UNIVERSIDADE FEDERAL DE ALAGOAS
INSTITUTO DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM
COMPUTACIONAL DE CONHECIMENTO

MARLOS TACIO SILVA

MODELOS PARA A CONSTRUÇÃO DE SISTEMAS TUTORES MULTIAGENTES

Maceió
2012

MARLOS TACIO SILVA

MODELOS PARA A CONSTRUÇÃO DE SISTEMAS TUTORES MULTIAGENTES

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Curso de Mestrado em Modelagem Computacional de Conhecimento do Instituto de Computação da Universidade Federal de Alagoas.

Orientador:
EVANDRO DE BARROS COSTA

Maceió
2012

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico
Bibliotecária Responsável: Helena Cristina Pimentel do Vale

S586m Silva, Marlos Tacio.
Modelos para construção de sistemas tutores multiagentes / Marlos Tacio.Silva. – 2012.
117 f. : il.

Orientador: Evandro de Barros Costa.
Dissertação (mestrado em Modelagem Computacional de Conhecimento) – Universidade Federal de Alagoas. Instituto de Computação. Maceió, 2012.

Bibliografia: f. 115-117.

1. Sistemas tutores multiagentes. 2. Modelo Mathema. 3. Sistema de autoria. 4. Rede de Petri. 5. Ontologia. I. Título.

CDU: 004.78:37.018.43

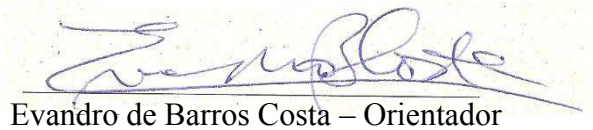
MARLOS TACIO SILVA

MODELOS PARA A CONSTRUÇÃO DE SISTEMAS TUTORES MULTIAGENTES

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Curso de Mestrado em Modelagem Computacional de Conhecimento do Instituto de Computação da Universidade Federal de Alagoas, aprovada pela comissão examinadora que abaixo assina.

Aprovado em 4 de Junho de 2012.

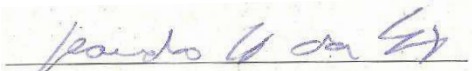
BANCA EXAMINADORA



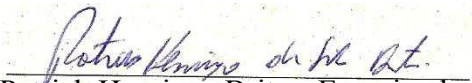
Evandro de Barros Costa – Orientador
Instituto de Computação
Universidade Federal de Alagoas

Frederico Luiz Gonçalves Freitas – Examinador
Centro de Informática
Universidade Federal de Pernambuco

Hyggo de Oliveira Almeida – Examinador
Departamento de Sistemas e Computação
Universidade Federal de Campina Grande



Leandro Dias da Silva – Examinador
Instituto de Computação
Universidade Federal de Alagoas



Patrick Henrique Brito - Examinador
Instituto de Computação
Universidade Federal de Alagoas

Dedico estre trabalho à minha família que sempre esteve comigo tanto nos bons quanto nos maus momentos.

AGRADECIMENTOS

Para toda a minha família que esteve comigo em todos os momentos da minha vida.

Ao meu orientador e amigo, professor Evandro, que soube ser bastante paciente comigo. Obrigado professor, jamais esquecerei do senhor.

Aos meus companheiros do Tips que me ajudam sempre.

Aos companheiros do Grow com quem passei grande parte da minha vida acadêmica.

Ao Vitor, secretário da Pós, que me ajudou muito no decorrer do curso.

Ao Marcelo, secretário do IC, que vem me ajudando desde os tempos da graduação.

“Com grande poderes vêm grandes responsabilidades.”

(Tio Ben)

RESUMO

Este trabalho se insere na linha de pesquisa Modelos Computacionais em Educação do Programa de Pós-graduação Interdisciplinar em Modelagem Computacional de Conhecimento, observando-se que um dos grandes desafios da área de Sistemas Tutores Inteligentes continua sendo abordar adequadamente a complexidade inerente à construção desses sistemas. Nesse contexto, pode-se abordar três aspectos relacionados a esta questão, a saber: (1) carência de diretrizes que guiem os construtores (i.e., autores e desenvolvedores) envolvidos no processo de construção dos ambientes; (2) a lacuna conceitual entre o conhecimento do autor e as ferramentas disponíveis para a construção do sistema; e (3) falta de uma arquitetura de *software* flexível e adequada para o desenvolvimento de entidades de *software* inteligentes. Assim, o presente trabalho tem o objetivo de apresentar uma sistemática, dotada de modelos, para auxiliar na construção de Sistemas Tutores Multiagentes baseados na arquitetura *Mathema*. Do ponto de vista do autor, essa sistemática visa auxiliar na modelagem do conhecimento do domínio via uma estrutura de grafo. A partir dessa estrutura deriva-se uma rede de Petri, para verificação tanto de propriedades estruturais quanto comportamentais, e uma base de conhecimento, que irá ser operacionalizada por um planejador pedagógico. Do ponto de vista do desenvolvedor, essa sistemática visa utilizar a estrutura de grafo definida pelo autor para identificar um conjunto de agentes tutores e, a partir daí, construir efetivamente tais agentes com base em uma arquitetura de *software* mais flexível. Para a avaliação empírica da proposta desenvolveu-se um estudo de caso que consiste na estruturação de um curso de Ciência da Computação. Além disso, foram desenvolvidos mais estudos específicos, um no contexto de Lógica Computacional e outro no contexto de Aprendizagem de Máquina. Esses estudos mostraram a viabilidade da utilização da proposta, conseguindo obter resultados satisfatórios nas soluções apresentadas para responder as questões de pesquisa abordadas.

Palavras chave: Sistemas Tutores Multiagentes; Modelo Mathema; Sistemas de Autoria; Redes de Petri; Ontologias.

ABSTRACT

This work is situated in an interdisciplinary research program on computational modeling of knowledge, focusing on one challenge in the field of Intelligent Tutoring Systems with respect to manage the complexity involved in effectively building such systems. In this context, three aspects related to the mentioned challenge were addressed: (1) lack of concrete guidelines to be used by the involved actors (i.e., authors and developers) in the process of building this environments; (2) conceptual lack between the author's knowledge and the available tools for that end; and (3) lack of a flexible and adequate software architecture for building intelligent software entities. Thus, this work aims to present a systematic approach with models to help the construction of Multiagent Tutoring Systems based on Mathema's architecture. From the author point of view, this systematic aims to modeling a given domain via a graph structure. Based on this structure we derive a Petri net, to check both structural and behavioral properties and a knowledge base, which will be operated by a pedagogical planner. From the viewpoint of the developer, this systematic aims to use the graph structure defined by the author to identify a set of tutor agents and, thereafter, builds these agents based on flexible software architecture. For the empirical evaluation, we develop a case study consisting in structuring a course of Computer Science. In addition, more specific studies were developed, one in the context of Computational Logic and another in the context of Machine Learning. These studies demonstrate the feasibility of using the proposal, obtaining satisfactory results in the solutions presented to answer the research questions addressed.
Key-words: Multiagent Tutoring Systems; Mathema's Model; Authoring Systems; Petri nets; Ontologies.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE DEFINIÇÕES

LISTA DE EXEMPLOS

LISTA DE ABREVIATURAS E SIGLAS

AC	Auto Conhecimento
AG	Agente Gerenciador
AH	Aprendiz Humano
AT	Agente Tutor
CS	Conhecimento Social
FIPA	*Foundation for Intelligent Physical Agents*
SATA	Sociedade de Agentes Tutores Autônomos
SEH	Sociedade de Especialistas Humanos
STI	Sistema Tutor Inteligente
STM	Sistema Tutor *Mathema*

LISTA DE SÍMBOLOS

G	Grafo que estrutura um *curriculum* do *Mathema*
U	Conjunto de unidades pedagógicas de um *curriculum* do *Mathema*
T	Conjunto que define as relações taxonômicas entre as unidades pedagógicas de um *curriculum* do *Mathema*
R	Conjunto que define as relações de ordem entre as unidades pedagógicas de um *curriculum* do *Mathema*
E	Conjunto de estados das unidades pedagógicas de um *curriculum* do *Mathema*
u_0	Unidade raiz da estrutura de *curriculum* do *Mathema*
RP	Rede de Petri
L	Conjunto de lugares de uma rede de Petri
Tr	Conjunto de transições de uma rede de Petri
Ar	Conjunto de arcos de uma rede de Petri
P	Peso associado a um determinado arco
M_0	Conjunto de marcações iniciais de uma rede de Petri
A	Agente *Mathema*
S	Conjunto de sensores de um agente *Mathema*
Pe	Conjunto de percepções de um agente *Mathema*
At	Conjunto de atuadores de um agente *Mathema*

SUMÁRIO

1. INTRODUÇÃO

O presente trabalho situa-se na linha de Modelos Computacionais em Educação, que no contexto desta dissertação envolve as áreas de Informática na Educação, Inteligência Artificial, Verificação Formal e Engenharia de *Software*. Nessa perspectiva, o trabalho descrito nesta dissertação é o resultado de um estudo sobre a concepção de Sistemas Tutores Inteligentes (STIs) numa abordagem multiagentes. Com isso, pretende-se uma contribuição à pesquisa no domínio da construção de tais sistemas tomando como base o modelo *Mathema* [1]. Neste capítulo apresenta-se um panorama geral do trabalho em questão, iniciando-se com uma contextualização e fixação do foco da pesquisa, delineando a abrangência do tema a ser estudado. Além disso, apresentam-se os problemas observados no tema em pauta, conectando-os com os objetivos do trabalho. Por fim, apresenta-se a estrutura do documento com um breve relato dos capítulos que se seguem.

1.1. Contextualização da Pesquisa

Sistemas Tutores Inteligentes são sistemas de *software* que têm como objetivo possibilitar o aprendizado individualizado de um estudante sobre um determinado domínio de conhecimento. Basicamente, visa-se prover sistemas que se adaptem as características do estudante, ou seja, busca-se a personalização do processo de ensino/aprendizagem [2]. No que tange a personalização, tais sistemas tomam como base informações de três fontes distintas, a saber [3]: (1) conhecimento sobre o que ensinar (i.e., modelo de domínio); (2) conhecimento sobre como ensinar (i.e., modelo pedagógico); e (3) conhecimento sobre para quem ensinar (i.e., modelo do estudante). Nesse sentido, os STIs ganharam grande importância ao longo dos anos e migraram dos laboratórios de pesquisa para as aplicações comerciais [4] [5] [6] [7].

No entanto, tais sistemas são complexos[1] e, portanto, geram um alto custo no seu desenvolvimento. Tal complexidade está relacionada principalmente a quantidade e

[1] O termo complexidade se refere à dificuldade inerente à construção de tais sistemas.

diversidade de conhecimento que é necessária para a sua efetiva construção. Particularmente nesse ponto, o modelo *Mathema* visa diminuir essa complexidade de construção investindo em uma abordagem multiagentes [8] [9]. Nessa abordagem, cada agente pode ser visto como um tutor em particular e, desse modo, é possível dividir as responsabilidades entre os diversos tutores do sistema, diminuindo assim a sobrecarga de conhecimento sobre cada um.

Em essência, o modelo *Mathema* foi elaborado para ser uma resposta efetiva ao problema de adaptação dinâmica de um sistema tutor às características particulares de um dado estudante envolvido em situações de resolução de problemas, contando potencialmente com uma assistência personalizada[2]. Nesse sentido, tal modelo propõe, numa visão macro, uma solução que se inicia com uma proposta de abordagem de modelagem de conhecimento de domínio (em duas perspectivas relacionadas: ponto de vista e *curriculum*) e, a partir daí, o mapeamento dos modelos em um conjunto de agentes tutores.

Nesse contexto, o processo de construção de Sistemas Tutores *Mathema* (STM) envolve um conjunto de *stakeholders*, doravante denominados construtores de *Mathema*, que, nesse trabalho, foram reduzidos a:

- **Especialista**: detém o conhecimento do domínio a ser modelado, sendo aqui representado pelo papel do professor. Esse *stakeholder* tem a função de definir a estrutura do curso, ou seja, construir os tópicos, os problemas abordados e o suporte de conhecimento disponível aos mesmos;

- **Engenheiro de Conhecimento**: possui a habilidade de modelar o conhecimento de domínio, desempenhando o papel de explicitar e representar computacionalmente o conhecimento do especialista em uma base de conhecimento;

- **Engenheiro de *Software***: detém o conhecimento acerca do desenvolvimento do sistema, tomando decisões como, por exemplo, quantos e quais agentes serão construídos, quais mecanismos serão instanciados, entre outros. Além disso, cabe a esse ator o papel de prover novas funcionalidades ou de adequar antigas funcionalidades aos novos requisitos.

[2] Mais informações sobre o *Mathema* serão apresentadas no Capítulo 2.

Desta proposta derivam-se vários estudos específicos, tendo tal tema sido também alvo de outras pesquisas inspiradas no modelo *Mathema* (e.g., Silva et al. [10], de Almeida et al. [11], Frigo et al. [12] e Bittencourt et al. [13]), tal como será apresentado mais adiante no Capítulo 3. Tais investimentos visam enriquecer ainda mais o modelo original, propondo desde ferramentas de autoria até arcabouços de *software* para o desenvolvimento de STMs. Todavia, é importante frisar que mesmo com esses investimentos a construção de STMs ainda é um tema que conta com problemas em aberto.

1.2. Questões de Pesquisa

Como visto anteriormente, o escopo deste trabalho é o da construção de STMs, privilegiando os construtores de *Mathema*, ou seja, o Especialista, o Engenheiro de Conhecimento e o Engenheiro de *Software*. Entretanto, é importante ressaltar que a abordagem apresentada nessa dissertação focaliza o autor do sistema, representado aqui pelo Especialista, como figura basilar no processo de construção de STMs e, por conseguinte, apresenta uma maior ênfase nesse *stakeholder*. Nessa perspectiva, a pretensão de se construir esses sistemas levanta uma série de questionamentos. Contudo, o interesse global desta pesquisa está focalizado no questionamento a seguir:

Q0. Como diminuir a complexidade na construção de STMs?

A diminuição de complexidade concerne o provimento de condições que facilitem a construção do sistema. Diante disto, tal indagação pode ramificar-se nas seguintes questões:

Q1. Como auxiliar o Especialista na construção de STMs?

Com respeito ao Especialista, um dos problemas envolvidos é a dificuldade do mesmo em explicitar seu conhecimento, ou seja, a carência de soluções que o auxiliem a estruturar os recursos envolvidos no processo de ensino/aprendizagem. Tal dificuldade ocorre porque existe uma lacuna conceitual entre os modelos de autoria existentes e o conhecimento do autor, ou seja, grande parte dos modelos existentes não se mostra intuitiva para o autor. Outro problema reside na verificação do conhecimento

construído pelo mesmo, ou seja, na falta de mecanismos que verifiquem o planejamento instrucional definido pelo professor.

Q2. Como auxiliar o Engenheiro de Conhecimento na construção de STMs?
Com respeito ao Engenheiro de Conhecimento, um dos problemas diz respeito a como construir as bases com o conhecimento explicitado pelo especialista, ou seja, a carência de mecanismos que transformem a estruturação do domínio em bases de conhecimento.

Q3. Como auxiliar o Engenheiro de *Software* na construção de STMs?
Com respeito ao Engenheiro de *Software*, um dos problemas envolve a identificação dos agentes. Além disso, a lacuna existente entre as fases de especificação e implementação torna o processo de desenvolvimento do sistema uma tarefa complexa, tendo em vista a dificuldade em manter a conformidade entre o modelo especificado pelo Especialista.

Nessa perspectiva, é possível perceber diversos problemas envolvidos na concepção desses sistemas. Diante disto, este trabalho visa apontar um conjunto de soluções que têm o objetivo de diminuir esses problemas. Para tal, a seção que segue apresenta um conjunto de objetivos que têm o papel de definir os encaminhamentos desta pesquisa.

1.3. Objetivos da Pesquisa

Com base na problemática apresentada, o objetivo geral deste trabalho pode ser definido, tal como segue:

O0. Definir uma sistemática, dotada de modelos, que tem a finalidade de diminuir a complexidade envolvida na construção de STMs sob a perspectiva do Especialista, do Engenheiro de Conhecimento e do Engenheiro de *Software*.

Assim, tendo em vista o objetivo geral apresentado, foi concebido um conjunto de objetivos específicos no intuito de responder aos questionamentos levantados. Portanto,

espera-se que as respostas para os três questionamentos identificados anteriormente, possam conduzir para uma solução macro que responda a questão base. Desse modo, é possível ramificar o objetivo geral deste trabalho nos seguintes objetivos específicos:

O1. Definir uma sistemática para a construção de STMs.

A consecução desse objetivo remete a definição de um conjunto de passos que irá resultar em: (1) apontamento dos agentes que irão compor o sistema; e (2) construção da base de conhecimento que será utilizada como suporte para um planejador pedagógico.

O2. Definir um conjunto de modelos para a construção de STMs.

Aos modelos compete definir primeiramente uma estrutura conceitual para a especificação dos STMs e, além disso, uma contrapartida computacional para a efetiva construção desses sistemas. Sob essa perspectiva, dentre os modelos a serem definidos, é possível destacar: (1) o modelo de estruturação do *curriculum*, que visa definir a estrutura de conhecimento do domínio; (2) o modelo de verificação, que tem o objetivo de verificar tanto características estruturais como características comportamentais da estrutura de conhecimento construída; (3) modelo de agentes, que tem o intuito definir as características de um agente *Mathema*, bem como identificar os agentes que irão compor o sistema tomando como base a estrutura curricular construída.

Diante do exposto, o desafio dessa dissertação consiste em contribuir para o processo de construção de STMs, diminuindo, assim, a complexidade envolvida na sua criação. Desse modo, os modelos que serão desenvolvidos têm o objetivo de abranger os três *stakeholders* apontados anteriormente. Contudo, é importante ressaltar que a proposta aqui defendida apresenta maior ênfase no modelo de autoria, que no contexto dessa dissertação compreendem as tarefas do Especialista.

1.4. Organização do Documento

O conteúdo desta dissertação está estruturado em capítulos, sendo o presente capítulo o primeiro. Contudo, o trabalho também pode ser visto como um agregado constituído por três partes. A Parte I, que é composta pelos Capítulos 2 e 3, tem o objetivo de apresentar o referencial teórico, introduzindo conhecimentos de apoio à leitura do trabalho bem como um apanhado de propostas relacionadas. A Parte II, que é constituída pelos Capítulos 4 e 5, visa abordar o desenvolvimento do trabalho, descrevendo os modelos desenvolvidos, bem como o estudo de caso efetuado. Por fim, na Parte III, que é constituída pelo Capítulo 6, são apresentadas as conclusões, as principais contribuições do trabalho e os direcionamentos futuros.

Mais especificamente, o conteúdo dos capítulos pode ser definido com base nos seguintes propósitos:

- Capítulo 2 – MATHEMA: MODELO DE ESTRUTURAÇÃO DE CONHECIMENTO: descreve os conceitos relacionados ao tema deste trabalho, apresentando as características dos sistemas baseados no *Mathema*, dando destaque para a modelagem do conhecimento do domínio, bem como para a arquitetura multiagentes;

- Capítulo 3 – TRABALHOS RELACIONADOS: realiza um levantamento dos trabalhos relacionados a esta proposta, descrevendo suas características e limitações. Além disso, apresenta uma comparação entre as propostas relacionadas, destacando os requisitos atendidos por cada uma;

- Capítulo 4 – ABORDAGEM PROPOSTA: apresenta a sistemática e os modelos propostos para construção de STMs. De modo geral, serão apresentadas as definições de três modelos: (1) modelo de estruturação; (2) modelo de verificação; e (3) modelo de agentes;

- Capítulo 5 – AVALIAÇÃO: apresenta, primeiramente, um caso geral com a modelagem da grade do curso de Ciência da Computação da Universidade Federal de Alagoas. Esse estudo visa destacar o modelo de estruturação do curricular. Além disso, apresenta mais um caso na modelagem de um curso de Aprendizagem de Máquina, destacando tanto sua estruturação como a verificação da estrutura. Por fim, apresente um caso na modelagem de um curso de Lógica, enfatizando a verificação e a identificação dos agentes.

- Capítulo 6 – CONCLUSÕES E PERSPECTIVAS FUTURAS: apresenta as considerações finais do autor, sumarizando os objetivos alcançados. Além disso, destaca as limitações do trabalho e aponta direcionamentos futuros para a referida pesquisa.

2. MATHEMA: MODELO DE ESTRUTURAÇÃO DE CONHECIMENTO

O *Mathema* é um modelo conceitual para estruturação de conhecimento de STIs baseados em uma arquitetura multiagentes [1]. Tal modelo define uma estrutura de organização do conhecimento de domínio, bem como uma arquitetura de agentes tutores. Neste capítulo, apresenta-se uma visão geral acerca do modelo de estruturação de conhecimento do *Mathema*, bem como da sua arquitetura multiagentes.

2.1. Modelo de Estruturação de Conhecimento

O *Mathema* define um modo de visualizar um dado domínio e, a partir daí, organizá-lo e particioná-lo apropriadamente objetivando, em última instância, apoiar o processo de interação que ocorre entre as entidades que compõem o sistema. Para tal, o *Mathema* define um modelo de conhecimento de domínio segundo duas frentes: (1) visão externa; e (2) visão interna.

2.1.1. Visão Externa

De modo geral, a visão externa de um domínio, doravante denominado domínio alvo, corresponde à possibilidade de atribuir diferentes interpretações a este domínio, resultando numa decomposição do mesmo segundo diferentes pontos de vista. Desse modo, o domínio alvo pode ser abordado segundo uma visão tridimensional [8]:

- **Contexto**: define diferentes pontos de vistas associados a um determinado domínio alvo. Isto é, define diferentes interpretações que podem ser dadas a um determinado domínio, constituindo diferentes abordagens para um mesmo objeto de conhecimento;

- **Profundidade**: define algum tipo de refinamento na linguagem de percepção de um determinado contexto. Isto é, define a estratificação dos vários níveis de conhecimento do domínio alvo em questão;

- **Lateralidade**: define conhecimentos laterais ou pré-requisitos localizados fora do plano que caracteriza o domínio em questão, sendo, portanto, necessário para abordar um determinado subdomínio que foge do foco do domínio alvo em questão.

Essa visão tridimensional mostra a possibilidade de um domínio poder ser enfocado por uma visão contextual, sendo que esta visão pode vir acompanhada de alternativas de variação do ponto de vista de profundidade e lateralidade em relação a cada contexto escolhido no domínio.

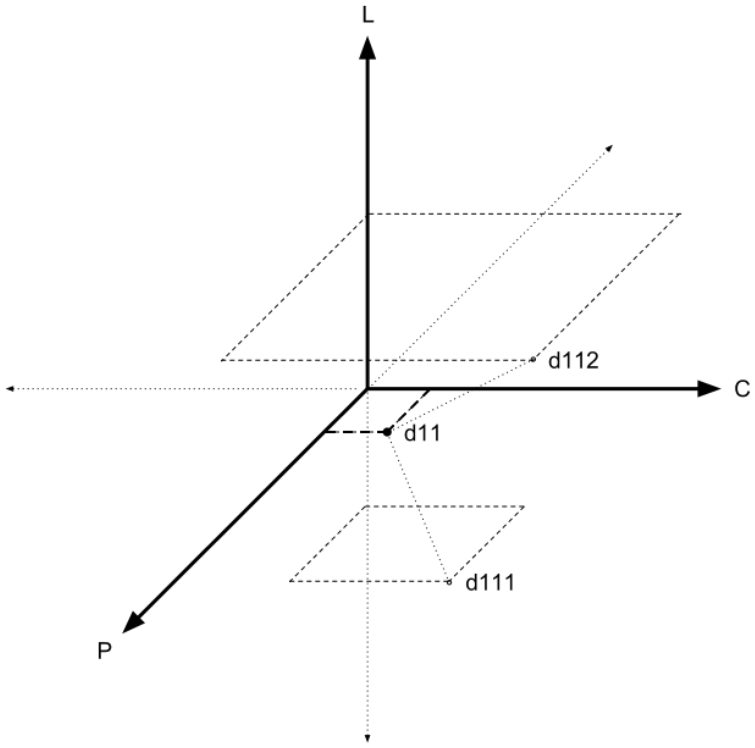


Figura 1: Visão tridimensional do modelo *Mathema*
Fonte: Adaptado de [1]

Como pode ser visto na Figura 1, um domínio alvo D é formado pelo plano definido pelas dimensões contexto e profundidade. Nesse plano é possível fixar um determinado contexto i e uma determinada profundidade j, servindo assim para associar e definir um

subdomínio desse plano. Lateralmente a esse plano estão situados os domínios laterais, logo para cada subdomínio fixado é possível derivar novos domínios alvo.

No intuito de clarificar as noções introduzidas anteriormente, segue um exemplo que ilustra a modelagem de um subdomínio da matemática, nesse caso em questão uma modelagem no domínio de frações. Tal domínio foi escolhido, primeiramente, porque o seu conhecimento é amplamente difundido e utilizado cotidianamente e, além disso, sua modelagem apresenta de forma adequada as noções de dimensão definidas pelo modelo *Mathema*.

Exemplo 1: Domínio de Fração

Seja D um domíno alvo que pretende modelar o conhecimento sobre frações, tem-se que de acordo com o modelo tridimensional do *Mathema* pode-se considerar para $D = Fração$, as seguintes dimensões:

- **Contextos**

  $C_1 = uma\ abordagem\ numérica$

  $C_2 = uma\ abordagem\ algébrica$

  As duas visões apresentadas (i.e., a abordagem numérica e a abordagem algébrica) se mostram equivalentes, ou seja, ambos os contextos admitem um mesmo conjunto de operações, existindo apenas uma diferença no modo como tais operadores são aplicadas. Desse modo, nota-se que esses contextos definem apenas modos diferentes de se focalizar um mesmo conhecimento.

- **Profundidades**

  $P_1 = Operações\ básicas$

  $P_2 = Exponenciação$

  $P_3 = Porcentagem$

  Com relação às profundidades, observa-se que essas representam uma estratificação da linguagem a ser utilizada, isto é, definem diferentes níveis de expressividade para a mesma. Assim cada nova profundidade pode ser vista como uma extensão da

linguagem anterior como, por exemplo, P_1, que compreende as quatro operações aritméticas básicas, é estendida pelas profundidades P_2 e, posteriormente, por P_3.

- **Lateralidades**

  $L_1 = Múltiplos\ e\ divisores$

  A lateralidade representa um conjunto de subdomínios que não pertencem ao plano em questão, mas que são necessários para a operacionalização do domínio alvo. Nesse domínio em questão, são necessários conhecimentos referentes ao uso de múltiplos e divisores (e.g., conceitos de múltiplos, divisores, números primos, decomposição, operações de máximo divisor comum e mínimo múltiplo comum). No entanto, tais conhecimentos fogem ao escopo do domínio de frações e, como tal, são modelados como lateralidades do domínio alvo. É importante frisar que cada lateralidade diz respeito a um novo domínio alvo que também é modelado utilizando a mesma estrutura tridimensional.

2.1.2. Visão Interna

Uma vez definida a visão externa, é preciso associar a cada visão, representada por um par formado por contexto e profundidade, um subdomínio e, a partir deste, definir as estruturas internas ao mesmo. Para tal, fixa-se cada subdomínio pertencente ao plano alvo em questão e define-se uma entidade, doravante denominada *curriculum*, para representar seus refinamentos necessários. Cada *curriculum* é constituído por três estruturas, a saber: (1) Unidades Pedagógicas; (2) Problemas; e (3) Conhecimento de Suporte. Estando cada uma delas especificadas a seguir [8]:

- **Unidade Pedagógica**: define um conjunto de tópicos associados a uma determinada visão do domínio. Tais tópicos se assemelham a um sumário e tem a função de organizar o *curriculum* como um todo. Além disso, essas unidades estão relacionadas segundo uma ordem definida com base em critérios pedagógicos (e.g., relações de pré-requisito);

- **Problema**: atividade avaliativa associada a uma determinada unidade pedagógica. Assim como as unidades pedagógicas, estão relacionados segundo uma ordem com base em critérios pedagógicos (e.g., nível de dificuldade);

- **Conhecimento de Suporte**: algum tipo de conhecimento que poderá ajudar o estudante a compreender determinado conceito ou resolver determinado problema (e.g., conceitos, exemplos, vídeo aulas, agentes humanos, agentes de *software* etc.).

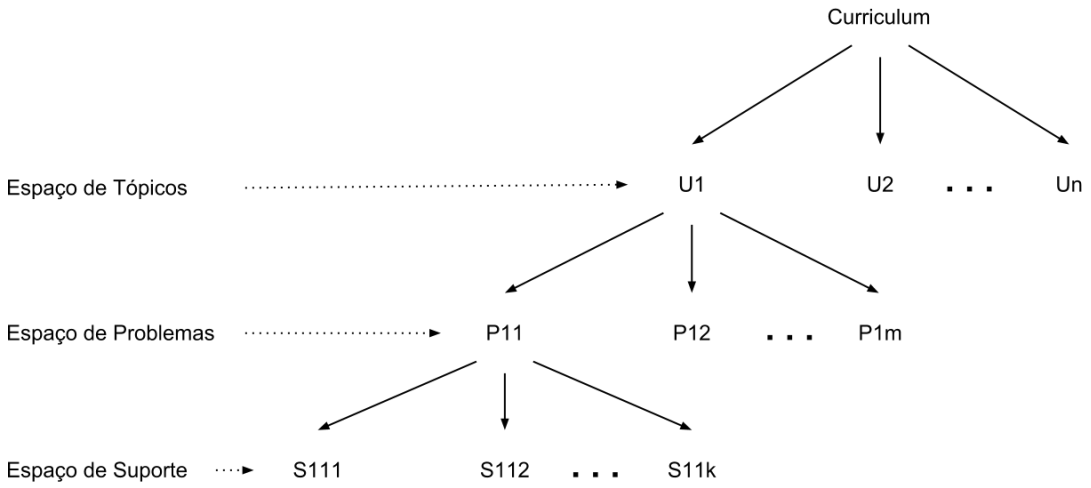


Figura 2: Estrutura do *curriculum* do *Mathema*
Fonte: Adaptado de [1]

A Figura 2 apresenta a estrutura pedagógica do conhecimento do domínio, nela podem ser vistos três planos fundamentais: (1) plano pedagógico; (2) plano de problemas; e (3) plano de suporte. No primeiro plano, também conhecido como espaço de tópicos, estão as unidades pedagógicas relacionadas segundo uma taxonomia, que tem a função de estruturar hierarquicamente as unidades, e um conjunto de relações de ordem, que tem a função de indicar o fluxo de execução das atividades pedagógicas. No segundo plano, também conhecido como espaço de problemas, encontram-se os problemas, que são as atividades pedagógicas no processo de ensino/aprendizagem. Tais problemas também estão relacionados com base em uma taxonomia e relações de ordem. Por fim, no último plano, também conhecido como espaço de suporte, encontram-se todo o conhecimento de apoio à resolução dos problemas como, por exemplo, conceitos, exemplos, dicas, pares complementares etc.

Visando exemplificar a estruturação do *curriculum*, prossegue-se com a modelagem do domínio de fração. Para tal, fixou-se o par $\langle C_1, P_1 \rangle$ para definir uma visão de operações aritméticas básicas em um contexto de frações numéricas e, a partir desta, construir a estrutura curricular correspondente. Para efeitos de exemplificação, será apresentada apenas a modelagem do espaço de tópicos desse *curriculum*.

Exemplo 2: *Curriculum* de Fração

Seja C um *curriculum* sob o qual se pretende estruturar o conhecimento de operações aritméticas básicas em frações numéricas, tem-se que de acordo com o modelo *Mathema* a estruturação desse *curriculum* é composta por um espaço de tópicos estruturado segundo uma taxonomia tal como segue:

- **Unidades Pedagógicas**

  $UP_1 = Frações\ numéricas.$

  O conceito mais geral do espaço de tópicos, aqui representado pela unidade UP_1. A partir desse conceito a estrutura de tópicos pode ser estratificada em subunidades organizadas segundo uma estrutura taxonômica. Desse modo, os conceitos partem de um conceito mais geral até os conceitos mais específicos, tal como a seguir.

  $UP_{1.1} = Conceituação\ e\ UP_{1.2} = Operações.$

  O segundo nível da taxonomia define os conceitos $UP_{1.1}$ e $UP_{1.2}$, que estruturam os tópicos de frações segundo uma visão conceitual e uma visual operacional. A visão conceitual apresenta basicamente um conjunto de definições e noções fundamentais para a compreensão de frações (i.e., as unidades $UP_{1.1.1}$, $UP_{1.1.2}$, $UP_{1.1.3}$, $UP_{1.1.4}$ e $UP_{1.1.5}$). A visão operacional apresenta duas ramificações (i.e., as unidades $UP_{1.2.1}$ e $UP_{1.2.2}$) que organizam o conjunto de operações em frações com base na aridade[3] de seus operadores.

[3] A aridade de uma função ou operação define o número de argumentos ou operandos da mesma.

$UP_{1.1.1} = Definição, UP_{1.1.2} = Tipologia, UP_{1.1.3} = Número\ Misto,$
$UP_{1.1.4} = Equivalência\ e\ UP_{1.1.5} = Propriedades.$

$UP_{1.2.1} = Unárias\ e\ UP_{1.2.2} = Binárias.$

Por fim, definem-se três tipos de operações unárias (i.e., as unidades $UP_{1.2.1.1}$, $UP_{1.2.1.2}$ e $UP_{1.2.1.3}$) e cinco operações binárias que dizem respeito à comparação entre frações e as operações aritméticas básicas (i.e., $UP_{1.2.2.1}$, $UP_{1.2.2.2}$, $UP_{1.2.2.3}$, $UP_{1.2.2.4}$ e $UP_{1.2.2.5}$).

$UP_{1.2.1.1} = Simplificação, UP_{1.2.1.2} = Redução\ e\ UP_{1.2.1.3} = Transformação.$

$UP_{1.2.2.1} = Comparação, UP_{1.2.2.2} = Adição, UP_{1.2.2.3} = Subtração,$
$UP_{1.2.2.4} = Divisão\ e\ UP_{1.2.2.5} = Multiplicação.$

Como dito anteriormente, a construção dos tópicos tem o objetivo de estruturar o *curriculum*, definindo uma organização tal como a de um livro ou de um curso. Nesse sentido, caberia ao espaço de problemas e ao espaço de suporte preencher tal estrutura com conteúdos e problemas a serem consumidos por um estudante no processo de ensino/aprendizado. É importante ressaltar que tanto a estruturação dos problemas quanto a estruturação do suporte se assemelham ao que foi apresenta no exemplo anterior.

2.2. Arquitetura do Sistema

A arquitetura multiagentes do *Mathema* consiste na integração entre entidades humanas e entidades de software. Assim, como pode ser visto na Figura 3, fazem parte do modelo *Mathema*, além da Sociedade de Agentes Tutores Artificiais (SATA), o Aprendiz Humano (AH), a Sociedade de Especialistas Humanos (SEH), o Agente de Interface, o Mediador Externo e o Agente de Manutenção [8].

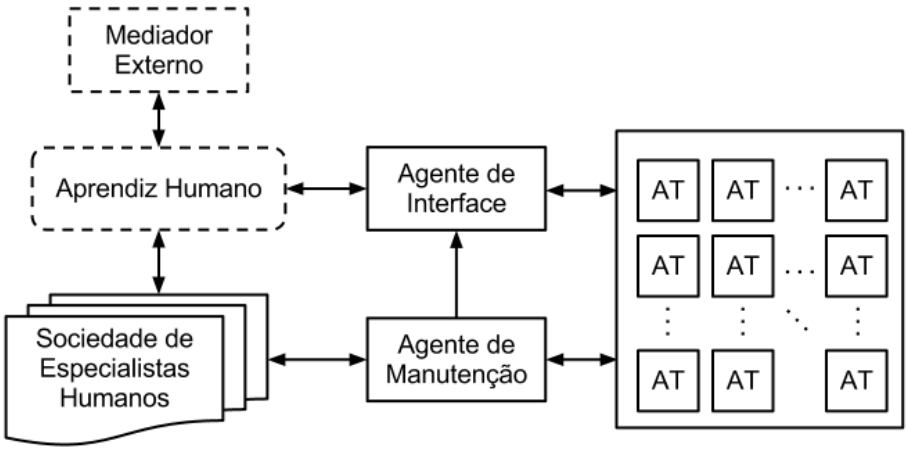


Figura 3: Arquitetura de um Sistema Tutor *Mathema*
Fonte: Adaptado de [1]

- **Aprendiz Humano**: agente interessado em aprender algo sobre um determinado domínio. Essencialmente, desempenhará o papel de agente ativo, envolvido em atividades baseadas na resolução de problemas de um dado domínio alvo. Para tal, contará com a assistência especializada de um Agente Tutor (AT);

- **Sociedade de Agentes Tutores Artificiais**: conjunto de agentes que podem cooperar entre si a fim de promover a aprendizagem de um dado aprendiz na atividade de resolução de problemas. Cada agente é especializado em determinada área de conhecimento do domínio alvo, ou seja, cada agente pode ser visto como um especialista em determinado tópico;

- **Sociedade de Especialistas Humanos**: conjunto de especialistas humanos que tem a função de criação e manutenção da SATA (i.e., com operações de inclusão, exclusão de agentes, bem como alterações no conhecimento dos agentes) e mais a disposição, em caso de uma falha mais crítica da SATA, de assistir os aprendizes. Assim, a SEH é responsável pela manutenibilidade das capacidades cognitivas da SATA;

- **Agente de Interface**: representa o elo entre o Aprendiz Humano e a SATA. Primeiramente, ele tem a responsabilidade de prover a comunicação entre agentes tutores e aprendizes. Além disso, é responsável por designar ao aprendiz um agente da SATA para agir como seu supervisor;

- **Agente de Manutenção**: representa o elo entre a SEH e a SATA, encarregando-se de prover uma interação entre os mesmos. Para isso, oferece os meios necessários para SEH realizar operações de manutenção sobre SATA. De certo modo, pode-se entender que este agente provê meios que facilitam o processo de aquisição de conhecimento de SATA;

- **Mediador Externo**: entidades humanas externas que desempenham o papel de motivar o Aprendiz a trabalhar no Sistema Tutor *Mathema*. Como exemplos de mediadores podem-se destacar um professor do aprendiz, seus colegas etc.

Assim, com base nessa arquitetura, um Sistema Tutor *Mathema* pode ser visto como dois ambientes distintos: (1) um ambiente de execução; e (2) um ambiente de construção/manutenção. Ao primeiro ambiente compete tutorar um estudante, provendo recursos personalizados a partir de um processo de ensino/aprendizagem guiado pelas características desse estudante. Ao segundo ambiente compete prover modelos e ferramentas que sejam capazes de construir o sistema como um todo. Nesse sentido, tal ambiente visa suprir as necessidades dos construtores de *Mathema* (e.g., Especialistas, Engenheiros de Conhecimento e Engenheiros de S*oftware*), provendo desde ferramentas de autoria, para a estruturação do curso, até arcabouços de *software*, para a construção dos agentes que comporão a SATA.

2.3. Agente *Mathema*

A arquitetura de um agente *Mathema* pode ser estruturada de modo a oferecer uma visão arquitetural em dois níveis de abstração distintos: (1) uma visão macro; e (2) uma visão micro.

2.3.1. Agente *Mathema*: Visão Macro

Diante de uma visão macro, um agente tutor pode ser estratificado em três componentes principais: (1) Sistema Tutor; (2) Sistema Social; e (3) Sistema de Distribuição. Como pode ser visto na Figura 4, os sistemas estão dispostos em uma arquitetura em camadas e, por conseguinte, existe uma interdependência entre as mesmas [8].

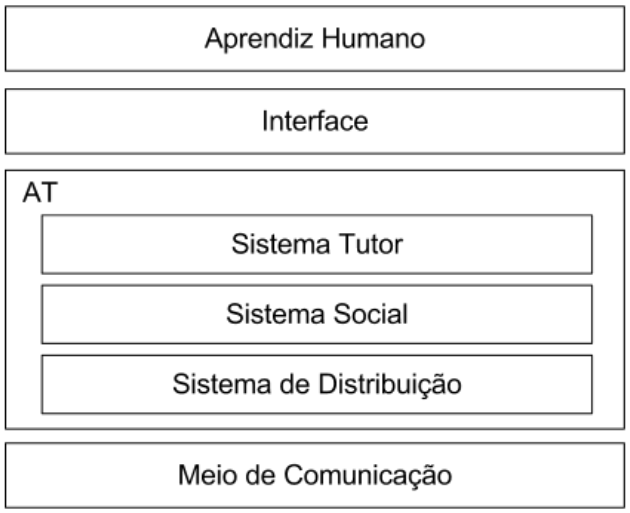


Figura 4: Visão macro de um agente *Mathema*
Fonte: Adaptado de [1]

- **Sistema Tutor**: responsável pela interação com o aprendiz humano, cabe a esse sistema executar atividades de tutoramento. Isoladamente, pode ser visto como um STI, por conseguinte nele fica armazenado todo o conhecimento que o agente possui sobre o tópico em questão;

- **Sistema Social**: responsável pelo comportamento cooperativo entre ATs, esse sistema é composto por bases de conhecimento e mecanismos de raciocínio acerca das habilidades de outros agentes, bem como de conhecimento de suas próprias habilidades;

- **Sistema de Distribuição**: responsável pela manipulação tanto das mensagens recebidas quanto das mensagens enviadas pelo agente. Além disso, tem a função de gerenciar a distribuição das mensagens para os módulos internos do agente tutor.

2.3.2. Agente *Mathema*: Visão Micro

Tendo em vista a arquitetura apresentada anteriormente, é possível definir uma visão interna a cada um dos sistemas apresentados. Essa visão tem o objetivo de destacar os módulos mais relevantes na viabilização da interação entre agentes, bem como destes com a sociedade de especialistas humanos e com o aprendiz humano. Como pode ser visto na Figura 5, cada sistema da arquitetura foi expandido e novos subsistemas foram destacados. Desse modo, pode-se destacar do Sistema Tutor [8]:

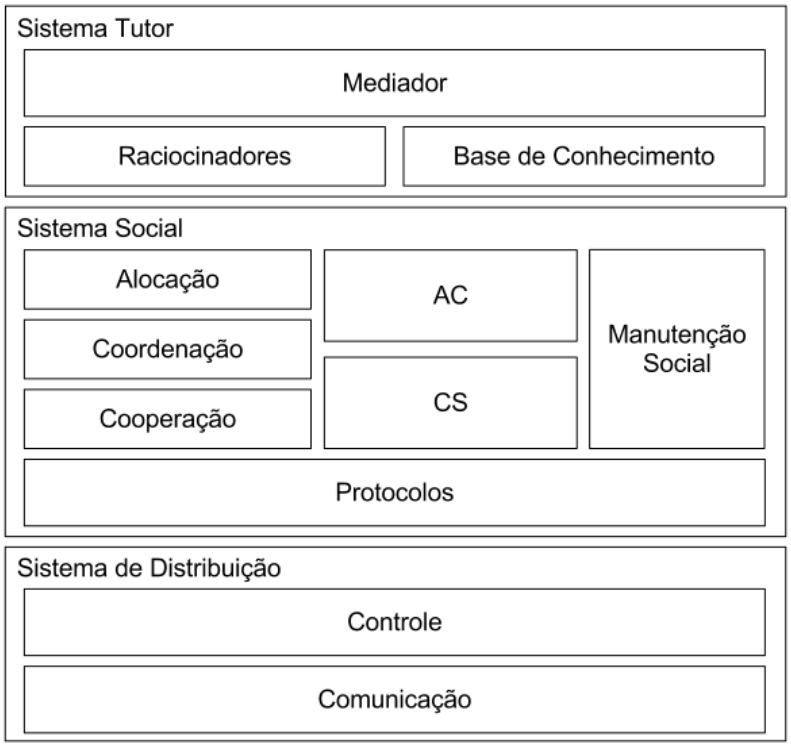


Figura 5: Visão micro de um agente *Mathema*
Fonte: Adaptado de [1]

- **Mediador**: tem como objetivo o controle geral da execução das funções pedagógicas no Sistema Tutor, ou seja, é o módulo responsável por interpretar as ações do aprendiz e, com base nestas ações, decidir qual o tipo de intervenção a ser realizada;

- **Raciocinadores**: envolve os mecanismos provedores das principais funções pedagógicas. Subdividem-se em três componentes: (1) módulo especialista; (2) módulo tutor; e (3) módulo do aprendiz. Primeiramente, o módulo especialista é

dotado de componentes que desempenham papéis de resolução de problemas, diagnósticos de soluções e remediação. Dando continuidade, o módulo tutor é responsável por selecionar os recursos pedagógicos inerentes a ele, levando em conta o contexto da interação. Por fim, o módulo de aprendiz tem a função de armazenar informações acerca do aprendiz, tais como acertos, erros, conhecimentos adquiridos etc.;

- **Bases de Conhecimento**: conhecimento relacionado à estrutura pedagógica a que se refere o agente, em outras palavras, é o conhecimento sobre o tópico em questão.

Com o módulo tutor os agentes possuem especialidades em resolver tarefas em domínios específicos. Contudo, certas tarefas necessitam do envolvimento de mais do que uma especialidade. Desse modo, cabe ao módulo de raciocínio realizar apenas as tarefas que a ele compete e, se for o caso, identificar as que não são de sua competência. Desse modo, tal módulo deve ser capaz de decompor uma tarefa para que a mesma possa ser enviada para as entidades que estão aptas a resolvê-la. Assim, o módulo Social apresenta, entre outras funções, a identificação das especialidades dos agentes da sociedade. Assim, é possível utilizar um módulo de alocação de tarefas para designar agentes para a resolução de uma tarefa. Mais especificamente, compreendem ao módulo Social os seguintes módulos [8]:

- **Autoconhecimento (AC)**: modelo que representa o que um agente sabe sobre si mesmo, isto é, quais são suas habilidades e conhecimentos. Este módulo é utilizado quando o agente precisa decidir se possui o conhecimento necessário para resolver uma determinada tarefa;

- **Conhecimento Social (CS):** modelo pelo qual o agente tem informações acerca do conhecimento dos outros agentes tutores da sociedade. Assim, quando um agente não é capaz de solucionar uma tarefa, este pode identificar agentes que estejam aptos a solucioná-la, criando uma cooperação entre agentes tutores;

- **Alocação**: responsável por selecionar os agentes aptos a solucionar uma determinada tarefa. Assim, dada uma lista de tarefas, este módulo deve ser capaz de designar um

agente para cada tarefa da lista. Após a alocação das tarefas este módulo repassa a lista de tarefas e agentes para o módulo de Coordenação;

- **Coordenação**: responsável por interpretar o grafo de tarefas e determinar qual tarefa deverá ser resolvida a cada momento. É importante frisar que o insucesso em uma determinada subtarefa causa um efeito em cascata que acarretará no insucesso do processo como um todo;

- **Cooperação**: responsável por promover a execução de uma tarefa. Assim, compete a esse módulo a resolução de conflitos na execução como, por exemplo, quando existe mais de um agente apto a solucionar a tarefa em questão. Por conseguinte, esse módulo deve ser capaz de ativar o protocolo mais adequado, dependendo da situação em que se encontre;

- **Manutenção Social**: responsável por prover mecanismos que mantenham a sociedade de agentes atualizada, a isto compreende tanto a entrada quanto a saída de agentes da sociedade e, além disso, compreende também a atualização do conhecimento social de cada agente;

- **Protocolos**: responsável pela criação e ativação de um diálogo com base nos protocolos disponíveis. Desse modo, tem-se que um protocolo define um comportamento de interação entre entidades.

Por fim, o Sistema de Distribuição fica encarregado de realizar as atividades de envio e recebimento de mensagens. Mais detalhadamente, este sistema pode ser definido pelos módulos que seguem [8]:

- **Controle**: responsável pela intermediação entre o Sistema Social e o Sistema de Distribuição. Basicamente, apresenta as seguintes funções: (1) encaminhamento das mensagens recebidas para as instâncias de diálogos apropriadas; (2) verificação da consistência das mensagens recebidas; e (3) eliminação de mensagens atrasadas;

- **Comunicação**: responsável pela distribuição e coleta de mensagens. Funciona como um mediador entre o Sistema de Distribuição e o meio de comunicação, provendo uma abstração para com o meio de comunicação utilizado.

2.4. Considerações Finais

A visão segundo as três dimensões e mais a organização interna de cada subdomínio definidas no modelo *Mathema*, são o alicerce para uma boa estruturação do conhecimento de um dado domínio. Esse tipo de estruturação do conhecimento servirá de base para a construção de uma sociedade de agentes tutores. Assim, essa organização e mais um mecanismos de interação, incorporado a um sistema multiagentes, visa tornar o ambiente mais adequado para promover a aprendizagem de forma personalizada e adaptativa. Além disso, a utilização de um sistema multiagentes se mostra adequada, sobretudo em domínios complexos, haja vista que cada agente pode ser visto como um sistema tutor particular. Desse modo, a sobrecarga acerca do conhecimento do domínio pode ser dividido entre os diversos agentes da sociedade. Por fim, a estrutura de agentes do *Mathema* permite que o conhecimento dos agentes possa ser combinado (i.e., permite a cooperação entre agentes) de modo a resolver problemas compostos como, por exemplo, em resoluções de problemas que envolvam as habilidades de mais de um agente tutor.

3. TRABALHOS RELACIONADOS

O trabalho de pesquisa desenvolvido nessa dissertação aborda algumas etapas presentes no processo de construção de STIs, tendo o modelo *Mathema* como referência, o qual é aqui assumido como representativo para sistemas com esse propósito de tutoria numa abordagem de agentes. Assim, os trabalhos selecionados como relacionados foram os que tomaram o *Mathema* como referência, tendo como foco algum aspecto envolvido na construção de sistemas tutores, tanto na perspectiva do autor, quanto do desenvolvedor. Entre eles, são destacados os que possuem uma maior proximidade em termos de propósitos específicos, sendo assim mais detalhados, e os que possuem pouca interseção com a abordagem aqui proposta. A discussão seguinte é feita de forma segmentada por tópicos comprometidos com as etapas que interessam ao contexto desse trabalho.

3.1. Abordagens com Foco no Autor

Nesta seção serão apresentados os trabalhos mais fortemente relacionados a esta dissertação. Estes trabalhos focalizam suas propostas no intuito de facilitar a construção do modelo de domínio do ponto de vista do autor. Nessa perspectiva, destacam-se os trabalhos de Costa [14] e principalmente o de Frigo [15].

3.1.1. Modelagem e Construção de uma Ferramenta de Autoria para um Sistema Tutorial Inteligente

O trabalho proposto por Costa [14] foca seus esforços no problema de autoria do modelo de domínio. Diante disto, o autor propõe a definição, a modelagem e o desenvolvimento de uma ferramenta de autoria (vide Figura 6). Essa ferramenta tem o objetivo de facilitar a estruturação do conhecimento de domínio, fornecendo para o especialista uma forma mais intuitiva de explicitar seu conhecimento. Esse trabalho está

inserido no contexto do projeto *Mathnet* e pretende prover uma ferramenta de autoria para que o Especialista/Engenheiro de Conhecimento possa criar um domínio seguindo os passos predeterminados por uma gramática de modelagem.

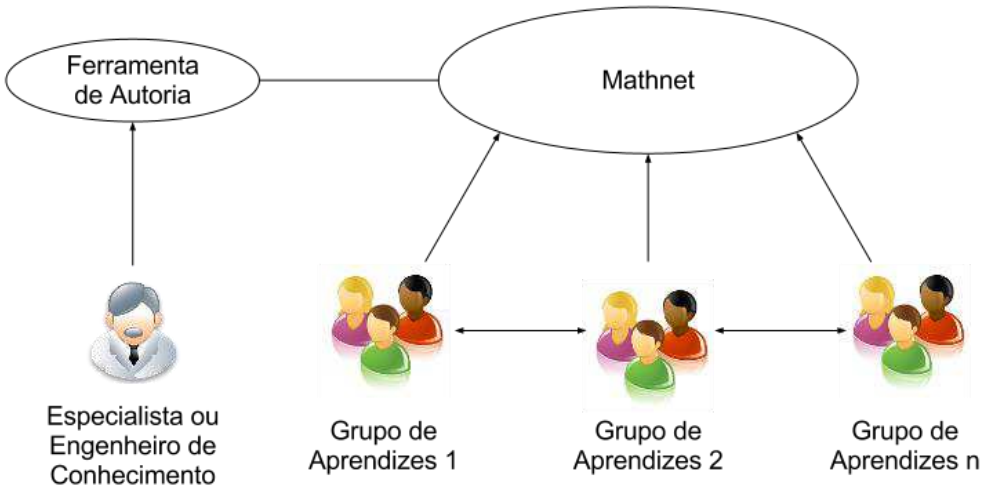


Figura 6: Ferramenta de autoria e o projeto Mathnet
Fonte: Extraído de [14]

Uma das grandes contribuições desse trabalho, além do desenvolvimento da própria ferramenta, reside na gramática criada para a especificação do domínio. A definição de uma gramática formaliza os conceitos envolvidos, além de prover uma base conceitual para que novas ferramentas possam ser construídas. Além disso, a proposta concebe uma ferramenta que visa cobrir todo o processo de construção do modelo de domínio, isto é, provê a definição tanto da visão externa quanto da visão interna do modelo *Mathema*.

No entanto, uma das limitações do trabalho consiste na simplificação da estruturação dos tópicos, não permitindo, por exemplo, a definição de taxonomias. Além disso, a proposta define apenas um tipo de relação de ordem, o pré-requisito. Desse modo, o modelo proposto não permite construções mais complexas e tampouco definições de estruturas hierárquicas. Dando continuidade, não são mencionados fatores importantes como, por exemplo, a verificação das estruturas criadas pelo professor ou até mesmo uma linguagem gráfica que auxilie o professor na estruturação do domínio.

3.1.2. Um Modelo para Autoria de Sistemas Tutores Inteligentes Adaptativos

O trabalho proposto por Frigo [15] foca seus esforços no problema de autoria e verificação do modelo de domínio. Nesse contexto, a autora do trabalho propõe a definição de uma ferramenta de autoria, dotada de uma linguagem visual apropriada, e a consequente verificação da estrutura construída utilizando redes de Petri orientadas a objetos [16]. Assim, como apresentado na Figura 7, é proposta uma linguagem visual baseada em grafo para a estruturação do *curriculum*. Desse modo, o professor pode utilizar as relações definidas para construir sequenciamentos mais complexos. Por conseguinte, é possível traduzir a estrutura construída em uma rede de Petri, para que sejam feitas as devidas verificações. Por fim, é proposta a transformação da rede em regras baseadas na máquina de inferência *Jess* [17]. Esse trabalho rendeu alguns desdobramentos (e.g., Frigo et al. [12] e Cardoso et al. [18]) que juntos representam a abordagem como um todo.

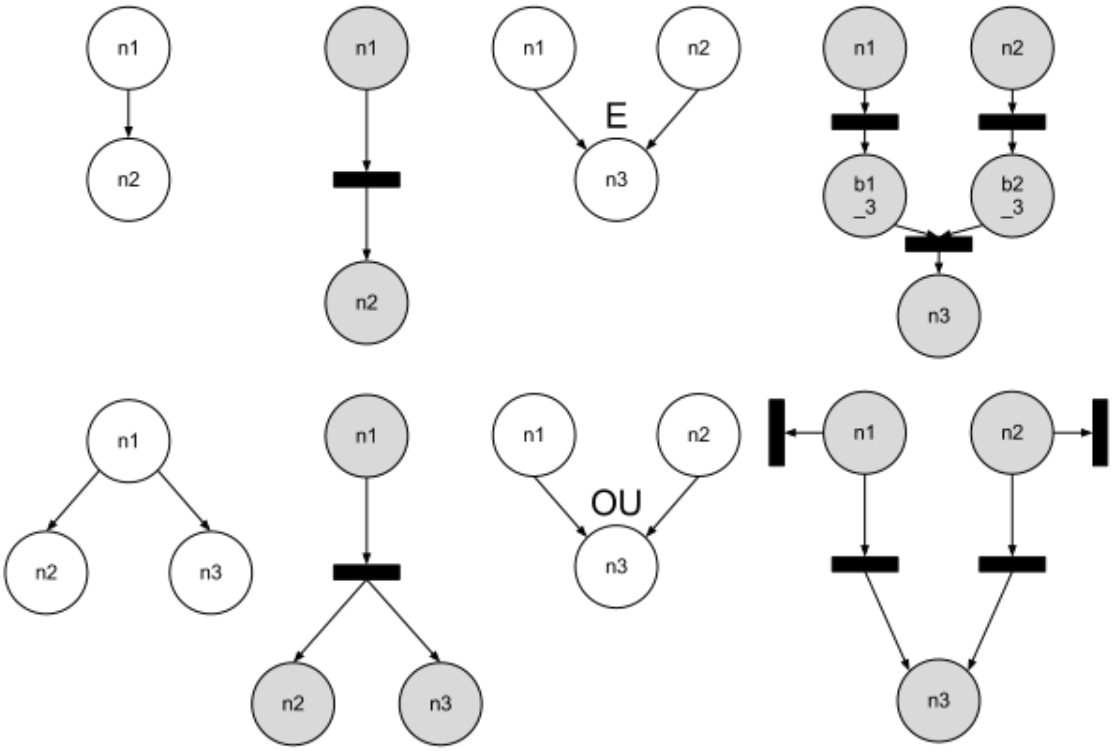


Figura 7: Topologia do grafo e rede de Petri resultante
Fonte: Extraído de [15]

As grandes contribuições desse trabalho consistem na definição de uma linguagem visual mais acessível ao professor, a utilização de um modelo de verificação e a consequente tradução da estrutura de *curriculum* em regras. Adicionalmente, a autora propõe o uso de

operadores lógicos para a construção de sequenciamentos mais complexos. Em contrapartida, essa abordagem não utiliza o conceito de relações taxonômicas, impossibilitando a estratificação do conhecimento em um determinado *curriculum*.

3.2. Abordagens com Foco em outros Atores

Nessa seção serão apresentados trabalhos que focam suas soluções em outros atores. Tais trabalhos se mostram mais como complementos a presente proposta, apresentando abordagens interessantes no que diz respeito ao modelo *Mathema*.

3.2.1. Aquisição de Conhecimento e Manutenção para uma Sociedade de Agentes Tutores Artificiais

O trabalho proposto por da Silva [19] foca na problemática relacionada à aquisição e manutenção dos agentes tutores, ou seja, na criação das bases de conhecimentos dos agentes e na eventual manutenção das mesmas. Desse modo, o autor propõe um ciclo de aquisição de conhecimento para a construção dos agentes, bem como um ambiente de manutenção que proveja o suporte a esse ciclo. Esse ciclo de aquisição é baseado nas seguintes tarefas: (1) organização externa, que é a criação dos subdomínios e o apontamento dos agentes que farão parte da sociedade; (2) estrutura pedagógica, definição da visão interna a cada subdomínio fixado; (3) modelagem conceitual de conhecimento dos subdomínios, que consiste na modelagem do agente em si; (4) geração das bases de conhecimento, geração de uma representação simbólica do conhecimento adquirido; e (5) criação dos agentes. Eventualmente, a fase de manutenção pode ser acionada para criar ou retirar agentes da sociedade, ou até mesmo atualizar o conhecimento de algum agente.

A presente proposta não foca diretamente na estruturação das unidades pedagógicas, pois sua preocupação maior é com o conhecimento dos agentes criados. Como tal, a proposta carece de modelos mais adequadas para a estruturação das unidades curriculares, ou seja, maiores facilidades para que o professor possa construir essas estruturas. Contudo, essa

proposta apresenta resultados significativos na manutenção das bases de conhecimento, provendo inclusive mecanismos de verificação de regras, visando encontrar redundâncias, circularidades, conflitos etc.

3.2.2. Compor – Desenvolvimento de *Software* para Sistemas Multiagentes

O trabalho proposto por de Almeida [20] foca na construção de sistemas multiagentes e no problema de evolução dos mesmos. Nesse contexto, o autor propõe uma metodologia, dotada de modelos e ferramentas, que auxiliem o Engenheiro de *Software* no processo de desenvolvimento de *software* orientado a agentes. Entre as contribuições da abordagem que ficou conhecida como COMPOR podem-se destacar uma metodologia, um modelo de componentes, um arcabouço de desenvolvimento e um ambiente de execução.

Esse trabalho se mostra interessante, pois foca seu esforço em resolver uma problemática macro que é a construção de Sistemas Multiagentes tomando como base o modelo *Mathema*. Pode-se destacar também o modelo de contêineres e componentes funcionais que criam uma abstração para o referenciamento de funcionalidades, eliminando a utilização de referências explicitas e garantindo maior flexibilidade na remoção e inserção de novos serviços.

3.2.3. Plataforma para Construção de Ambientes Interativos de Aprendizagem Baseados em Agentes

O trabalho proposto por Bittencourt [21] trata do problema do desenvolvimento de Ambientes Interativos de Aprendizagem. Diante disto, o autor foca sua solução na figura do Engenheiro de *Software* e, assim, propõe a construção de um arcabouço [22] para o desenvolvimento de ambientes interativos baseados no *Mathema*. A plataforma é composta basicamente pela interação entre ferramentas e um conjunto de agentes. Desse modo, as ferramentas provêm funcionalidades básicas (e.g., fórum, e-mail, enquete etc.) enquanto que os agentes provêem as funcionalidades de tutoramento e gerenciamento da plataforma. Além

disso, a camada de configuração tem o objetivo de prover uma porta de entrada para que autores possam configurar a plataforma.

Um dos diferenciais dessa proposta consiste da utilização de ontologias para a descrição dos modelos (e.g., modelo do estudante, modelo de domínio, modelo pedagógico) que compõem a plataforma [23]. Além disso, a infraestrutura de agentes é baseada no arcabouço JADE [24] e, como tal, segue as normas definidas pela *Foundation for Intelligent Physical Agents* (FIPA)[4].

3.3. Considerações Finais

Como apresentado no decorrer do presente capítulo, existem diversas abordagens que tomam como base a estrutura definida pelo *Mathema*. Nesse contexto, a presente seção visa encerrar o presente capítulo com uma discussão acerca dos trabalhos apresentados, apontando futuros desdobramentos que serão apresentados posteriormente no Capítulo 4. Assim, dentre as propostas apresentadas, há que se destacarem as abordagens direcionadas ao autor do sistema, representado aqui pelo papel do Especialista. Nesse sentido, as abordagens apresentados por Costa [14] e Frigo [25] se mostram fortemente relacionadas à proposta dessa dissertação, pois ambas propõem mecanismos para a estruturação do *curriculum*.

Nesse contexto, é importante ressaltar, primeiramente, o modelo de verificação proposto por Frigo [25]. Tal modelo se mostra de grande importância na estruturação curricular, haja vista que provê mecanismos para verificação das estruturas especificadas pelo especialista. Além disso, esse trabalho propõe uma contrapartida visual, ou seja, uma notação gráfica para a estruturação do *curriculum.* Do trabalho proposto por Costa [14], é possível destacar a definição de uma linguagem para estruturação do *curriculum*, ou seja, uma notação textual que pode ser facilmente operacionalizada por um compilador. No entanto, ambas as propostas realizam algumas simplificações no modelo original tal como definido por de Barros Costa [1] como, por exemplo, a definição de taxonomias. Esse tipo de simplificação impossibilita a estratificação do conhecimento, diminuindo assim a expressividade do modelo

[4] Mais informações em http://www.fipa.org/.

e tornando-o menos compatível com a visão utilizada na grande maioria dos cursos desenvolvidos atualmente, onde se utiliza uma estrutura de árvore para organização dos tópicos. Além disso, ambas as propostas carecem de mais detalhamento nas definições de seus modelos e, nesse sentido, necessitam de formalizações mais adequadas.

Por outro lado, outras propostas focalizam o processo de aquisição de conhecimento, tarefa que compete ao Engenheiro de Conhecimento. Nesse sentido, os trabalhos propostos por Silva [19], Bittencourt [21] e Frigo [25], apresentam propostas interessantes no que dize respeito ao papel do Engenheiro de Conhecimento. A proposta apresenta por Silva [19] apresenta um ciclo de aquisição de conhecimento visando a estruturação curricular e a construção dos agentes. Nessa mesma perspectiva, as propostas de Bittencourt [21] e Frigo [25] utilizam ontologias para descrever a estruturação curricular. A abordagem com ontologias se mostra bastante atual e apresenta ganhos como, por exemplo, o aumento da expressividade na descrição dos conceitos. Além disso, o uso de ontologias se mostra uma abordagem interessante, haja vista a integração com os agentes de *software* que irão compor a sociedade de agentes. Entretanto, as ontologias propostas são baseadas em simplificações do modelo *Mathema* e, como tal, sofrem dos mesmos problemas já citados anteriormente.

Dando continuidade, algumas propostas focalizam o processo desenvolvimento do sistema, tarefa que compete ao Engenheiro de *Software*. Nesse sentido, os trabalhos propostos por de Almeida [20] e Bittencourt [21] apresentam boas soluções do ponto de vista da Engenharia de *Software*. A solução proposta por de Almeida [20], apresenta uma nova estruturação para a sociedade de agentes baseada na estrutura curricular. Essa nova estruturação, baseada em uma estrutura de árvore, facilita a inserção e remoção de novos agentes e habilidades na sociedade, tornando o processo de evolução mais facilitado. O trabalho proposto por Bittencourt [21], apresenta uma arcabouço de *software* para a construção dos agentes da sociedade. Nesse sentido há que se destacar a integração com o arcabouço JADE e, consequentemente, a conformidade com as definições da FIPA. Todavia, tais propostas não apresentam facilidades para o autor, onerando a criação dos STMs do ponto de vista do especialista.

Finalizando, a proposta aqui defendida, tal como será posteriormente apresentada, focaliza seus esforços em contribuir com modelos que auxiliem na construção de STMs. No que tange o especialista, serão definidos modelos que visam prover uma estruturação mais de

acordo com o que fora proposto por de Barros Costa [1], bem como visa-se definir tanto uma notação textual quanto uma noção gráfica que auxiliem o autor na estruturação do *curriculum*. Além disso, pretende-se definir um modelo de verificação inspirado no trabalho apresentado por Frigo [25]. Com relação ao Engenheiro de Conhecimento, visa-se construir uma ontologia que tome como base o modelo de estruturação do *curriculum* definido. Por fim, pretende-se para o Engenheiro de Conhecimento uma revisão da arquitetura original do *Mathema*, tomando como base, primeiramente, uma adaptação do modelo de contêineres definido por de Almeida [20] e, além disso, uma integração com o arcabouço JADE.

4. ABORDAGEM PROPOSTA

Como apresentado no Capítulo 2 a concepção de Sistemas Tutores *Mathema* é uma tarefa que engloba um conjunto de etapas diversificadas. Por conseguinte, é preciso prover diretrizes que visam a definir quais etapas serão contempladas e, além disso, como estas etapas se relacionam no processo de engenharia do sistema. Desse modo, será apresentada, no presente capítulo, uma sistemática para a especificação de STMs, bem como serão definidos um conjunto de modelos que têm o objetivo de auxiliar os *stakeholders* na estruturação e verificação do conhecimento de domínio. Por fim, será apresentado um modelo de agentes e uma revisão na arquitetura de agentes proposta no *Mathema.*

4.1. Sistemática

Tendo em vista que o foco da presente pesquisa é o da concepção de STMs, resolveu-se focalizar os esforços em uma sistemática que contemple a estruturação do conhecimento de domínio a partir da visão interna, mais especificamente no plano das unidades pedagógicas (i.e., espaço de tópicos). Resolveu-se investir nessa simplificação porque é a partir das unidades que serão definidos os agentes que comporão SATA e, além disso, a estruturação dessas unidades será utilizada como suporte para a operacionalização de um planejador pedagógico. Adicionalmente, acredita-se que a construção desse plano seja um dos pontos cruciais na construção do sistema, pois envolve, entre outras coisas, a aquisição do conhecimento do especialista. Ressalta-se ainda, que os conceitos definidos para esse modelo de tópicos podem ser facilmente aplicados tanto no espaço de problemas como no espaço de suporte, uma vez que ambos baseiam-se em estruturas similares entre si.

Em vista do apresentado, a sistemática para a modelagem de STMs toma como ponto inicial a estruturação do conhecimento de domínio. Assim, como apresentado na Figura 8, o especialista define a estrutura de um *curriculum* a partir das unidades pedagógicas. Essa estruturação é representada por um grafo composto pela taxonomia das unidades, pelas relações de ordem entre estas unidades e pelos estados das mesmas. Dando continuidade, é

realizada a verificação de propriedades (e.g., ausência de ciclos na estrutura, não conectividade da estrutura, unidades não alcançáveis, estados não permitidos etc.) do grafo construído pelo especialista. Essa verificação é realizada utilizando-se redes de Petri coloridas [26].

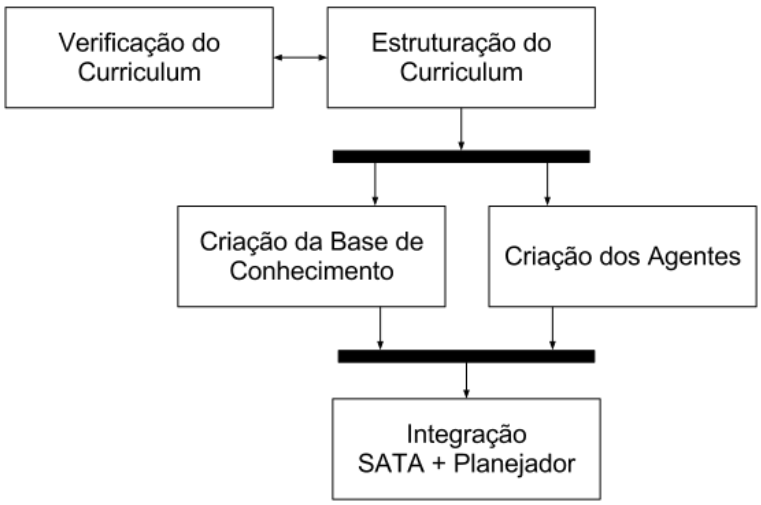


Figura 8: Sistemática para a concepção de STMs

Uma vez garantida a qualidade da estrutura, a sistemática se divide em dois ramos, que caminham em direção a objetivos complementares, a saber: (1) construção da base de conhecimento; e (2) identificação dos agentes de SATA. No ramo da esquerda visa-se caminhar em direção a uma base de conhecimento que será operacionalizada por um planejador pedagógico. No ramo da direita visa-se caminhar em direção a uma sociedade de agentes tutores, que terá a função de acompanhar o estudante no processo de ensino/aprendizagem. Por conseguinte, primeiramente é realizado o apontamento dos agentes tutores, tomando como base a estrutura curricular definida. A partir daí, a segunda etapa desse ramo tem o objetivo de construir efetivamente os agentes tutores e, para isso, utilizar-se-á uma revisão da arquitetura de agentes do *Mathema*.

Por fim, a última etapa visa a integração entre os agentes tutores e o planejador pedagógica. Juntas, essas duas entidades serão responsáveis pelo funcionamento do ambiente de execução do *Mathema*.

4.2. Modelo de Estruturação do *Curriculum*

O modelo de estruturação tem a função de organizar as unidades pedagógicas inerentes ao *curriculum* segundo relações taxonômicas e relações de ordem. Essa estruturação tem o objetivo de construir uma sequência lógica para que o sistema possa guiar o estudante no processo de ensino/aprendizagem. Nesse contexto, o presente modelo visa a definir as entidades e as relações pertinentes às mesmas, bem como uma linguagem para a modelagem dessas entidades. Adicionalmente, utiliza-se uma notação gráfica no intuito auxiliar o especialista na organização dessa estrutura. Por fim, define-se um modelo computacional para que tal estrutura possa ser operacionalizada por entidades de *software*.

4.2.1. Definição do Modelo

Basicamente, o modelo de estruturação do *curriculum* pode ser visto como um grafo composto por um conjunto de unidades pedagógicas. Cada unidade pode estar relacionada a outra com base em dois tipos de relacionamentos: (1) uma relação de taxonomia; e (2) uma relação de ordem. Formalmente, o grafo de unidades pedagógicas pode ser descrito tal como na Definição 1.

Definição 1: Seja G um grafo de entidades sobre a qual se pretende definir uma estruturação de acordo com a proposta do modelo *Mathema*, um enfoque para esse grafo pode ser definido por $G = (U, T, O, E, v_0)$. Assim, tem-se que:

i) A G associa-se um conjunto $U = \{u_0, \cdots, u_i, \cdots, u_{|U|}\}$, onde cada u_i representa o i-ésimo nó do grafo, que nesse contexto diz respeito a uma unidade pedagógica particular;

ii) A G associa-se um conjunto $T = \{(u_i, u_j) \mid u_i \in U \wedge u_j \in U \wedge u_i \neq u_j \wedge |T| = |U| - 1\}$, onde cada par $(u_i, u_j) \in T$ representa uma aresta direcionada que interliga o i-ésimo e

o j-ésimo nós do grafo, definindo, assim, uma relação taxonômica entre esses dois nós[5];

iii) A G associa-se um conjunto $O = \{(u_i, u_j) | \, u_i \in U \wedge u_j \in U \wedge u_i \neq u_j\}$, ode cada par $(u_i, u_j) \in O$ representa uma relação de ordem entre os i-ésimo e o j-ésimo nós do grafo[6];

iv) A G associa-se um conjunto $E = \{(u_i, e_i) | u_i \in U \wedge e_i \in \{0,1\} \wedge |E| = |V|\}$, representa o estado do i-ésimo nó do grafo, onde o estado 0 representa que o nó não foi concluído pelo estudante e o estado 1 representa que o nó foi concluído pelo estudante;

v) A G associa-se um nó $u_0 \in U$, que doravante será denominado nó raiz do grafo.

É importante ressaltar que, do ponto de vista taxonômico, o grafo segue uma estrutura semelhante a uma árvore e suas arestas representam uma relação taxonômica (i.e., relacionamento entre pai e filho) entre dois nós. Desse modo, o especialista pode definir uma estrutura hierárquica entre as diferentes unidades pedagógicas que compõem um curso, tal como a estrutura de um livro didático que é definida por capítulos, seções, subseções etc. Esse tipo de estruturação é amplamente conhecido e se mostra intuitivamente satisfatório para a estruturação do conhecimento de domínio.

Diante do exposto, a Definição 2 apresenta um conjunto de funções que formalizam essas estruturas hierárquicas.

Definição 2: Seja U o conjunto de nós do grafo de conceitos G, existe um conjunto de funções que definem uma relação taxonômica entre os nós adjacentes desse grafo (i.e., nós que estão ligados por uma mesma aresta $(u_i, u_j) \in T$). Tais relacionamentos podem ser definidos tal como segue:

[5] Mais informações serão apresentadas na Definição 2.
[6] Mais informações serão apresentadas na Definição 3.

i) Dado um nó u_i, existe uma relação de paternidade entre este nó (a.k.a., nó filho) e um nó u_j (a.k.a., nó pai) definido por

$pai: U \rightarrow U,$
$pai(u_j) = \{u_i \mid u_i \in U \ \wedge u_j \in U \wedge (u_i, u_j) \in T\};$

ii) Dado um nó u_i, existe uma relação de paternidade entre este nó (a.k.a., nó pai) e um conjunto de nós pertencentes a U' (a.k.a., nós filhos) definido por

$filhos: U \rightarrow U,$
$filhos(u_i) = \{u_k \mid u_k \in U' \wedge u_i \in U \wedge (u_i \times U') \subseteq T\};$

iii) Dado um nó u_i, existe uma relação de fraternidade entre este nó e um conjunto de nós pertencentes a U' (a.k.a., nós irmãos) definido por

$irmaos: U \rightarrow U,$
$irmaos(u_i) = \{u_l \mid u_l \in U' \wedge u_i \in U \wedge U' = filhos(\, pai(u_i)\,) - u_i\}.$

Além da estrutura taxonômica, o grafo de unidades é composto também por um conjunto de relações de ordem. Essas relações têm o objetivo de definir um fluxo de exploração entre os nós do grafo, ressaltando-se, no entanto, que só é possível estabelecer um relacionamento de ordem entre nós irmãos, ou seja, entre nós que estejam sob a tutela de um mesmo nó pai. Desse modo, tipos de relações de ordem podem ser definidas, tais como: (1) relação obrigatória, que define um fluxo de exploração obrigatório entre um nó e outro; (2) relação opcional, que define um fluxo de exploração opcional entre um nó e outro; e (3) relação alternativa, que define um fluxo alternativo entre dois nós. Todavia, é importante frisar que novos tipos de relações de ordem podem ser definidos, sendo os tipos de relações aqui definidos apenas um pequeno conjunto que poderá futuramente ser expandido.

Assim, a partir desses conceitos, é possível formalizar os relacionamentos de ordem presentes na estrutura de *curriculum* com base na Definição 3.

Definição 3: Seja R o conjunto de relações de ordem do grafo G, tal conjunto pode ser subdividido em três tipos de relações tal como segue:

i) Dado um nó u_i, existe uma relação de obrigatoriedade entre esse e os nós pertencentes ao conjunto U' com base na seguinte construção

$$Obg = \{(u_i, u_j) | \, u_i \in U \wedge u_j \in U' \wedge U' \subseteq irmaos(u_i) \wedge |U'| \geq 1\};$$

ii) Dado um nó u_i, existe uma relação de opcionalidade entre este nó e os nós pertencentes ao conjunto U' com base na seguinte construção

$$Opc = \{(u_i, u_j) | \, u_i \in U \wedge u_j \in U' \wedge U' \subseteq irmaos(u_i) \wedge |U'| \geq 1\};$$

iii) Dado um nó u_i, existe uma relação de alternância entre este nó e os nós pertencentes ao conjunto U' com base na seguinte construção

$$Alt = \{(u_i, u_j) | \, u_i \in U \wedge u_j \in U' \wedge U' \subseteq irmaos(u_i) \wedge |U'| \geq 2\}.$$

No intuito de compreender melhor a semântica das relações de ordem, duas funções devem ser definidas. A primeira, tal como pode ser visto na Definição 4, define uma função para que se possa obter o estado de um determinado nó do grafo. A segunda, tal como pode ser visto na Definição 5, define uma função de interpretação da relação de ordem, ou seja, retorna o estado desse relacionamento.

Definição 4: Seja U o conjunto de nós do grafo G, o estado de cada nó $u_i \in U$ pode ser obtido a partir da função

$$estado: U \rightarrow \{0, 1\},$$
$$estado(u_i) = \{e_i | (u_i, e_i) \in E\}.$$

Definição 5: Seja R o conjunto de relações de ordem do grafo G, o estado de cada relação $r_i \in R$ pode ser obtido a partir de função de uma função de interpretação

$$I: R \rightarrow \{0, 1\},$$
$$I(r_i) = 1, caso\ a\ relação\ esteja\ em\ um\ estado\ válido\ e$$
$$I(r_i) = 0, caso\ a\ relação\ esteja\ em\ um\ estado\ inválido.$$

Uma vez apresentados os conceitos de estado de um nó e interpretação de uma relação de ordem, é possível definir a semântica dos três tipos de relações de ordem previamente descritos na Definição 3. Assim, como apresentado na Definição 6, a semântica das relações de ordem pode ser definida por:

- **Obrigatório**: define uma relação de obrigatoriedade dos nós destino para com o nó de origem, ou seja, os nós de destino devem estar no mesmo estado do nó origem;

- **Opcional**: define uma relação de opcionalidade dos nós de destino para com o nó de origem, ou seja, caso o nó de origem esteja concluído, os nós de destino podem ou não estar concluído e, caso o nó de origem não esteja concluído, nenhum dos nós destino deve estar concluído;

- **Alternativo**: define uma relação de alternância dos nós de origem para com o nó origem, ou seja, caso o nó de origem esteja concluído, pelo menos um dos nós de destino deve estar concluído e, caso o nó de origem não esteja concluído, nenhum dos nós de destino deve estar concluído.

Definição 6: Seja R o conjunto de relações de ordem do grafo G, tal conjunto pode ser subdividido em três tipos de relações tal como segue:

i) Dado uma relação de ordem $r \in Obg$ a interpretação para essa relação pode ser definida por

$$I(r) = \{1 \mid r = (u_i \times U') \subseteq Obg \wedge (\forall u \in U', estado(u) = estado(u_i)\ )\ \}\ e$$
$$I(r) = 0, nos\ demais\ casos;$$

ii) Dado uma relação de ordem $r \in Opc$ a interpretação para essa relação pode ser definida por

$I(r) = \{0 \mid r = (u_i \times U') \subseteq Opc \wedge (\exists u \in U, estado(u) = 1 \wedge estado(u_i) = 0)\}$
$I(r) = 1, nos\ demais\ casos;$

iii) Dado uma relação de ordem $r \in Alt$ a interpretação para essa relação pode ser definida por

$I(r) = \{0 \mid r = (u_i \times U') \subseteq Alt \wedge (\exists u \in U', estado(u) = 1 \wedge estado(u_i) = 0)\},$
$I(r) = \{0 \mid r = (u_i \times U') \subseteq Alt \wedge (\forall u \in U', estado(u) = 0 \wedge estado(u_i) = 1)\}\ e$
$I(r) = 1, nos\ demais\ casos.$

A partir das relações de ordem é possível utilizar duas funções que visam obter os nós anteriores e os posteriores a um determinado nó do grafo, tal como apresentado na Definição 7. Semanticamente, os nós anteriores podem ser vistos como pré-requisitos de um determinado conceito, já os nós posteriores definem conceitos a serem vistos.

Definição 7: Seja R o conjunto de relações de ordem do grafo de G, existe um conjunto de funções que obtém tanto os nós posteriores quanto os nós anteriores de um dado nó $\boldsymbol{u_i}$:

i) Dado um nó u_i, existe um conjunto de nós posteriores ao mesmo que pode ser obtido a partir da função

$pos: U \rightarrow U,$
$pos(u_i) = \{u_j \mid u_i \in U \wedge u_j \in U' \wedge (u_i \times U') \subseteq R\};$

ii) Dado um nó u_i, existe um conjunto de nós anteriores ao mesmo que pode ser obtido a partir da função

$pre: U \rightarrow U,$
$pre(u_i) = \{u_k \mid u_i \in U \wedge u_k \in U' \wedge (U' \times u_k) \subseteq R\}.$

Por fim, se define o conceito de nó inicial, que é o nó que não possui nenhum pré-requisito (vide Definição 8), e nó final, que é o no que não possui nenhum nó posterior (vide Definição 9).

Definição 8: Seja U um conjunto composto por nós do grafo G, existe um subconjunto U' de nós (a.k.a., nós iniciais) que não possuem pré-requisitos. Tal subconjunto pode ser obtido a partir da função

$$iniciais: U \rightarrow U,$$
$$inciais(U) = \{u_i \mid u_i \in U \wedge pre(u_i) = \emptyset\}.$$

Definição 9: Seja U um conjunto composto por nós do grafo G, existe um subconjunto U' de nós (a.k.a., nós finais) que não possuem nós posteriores. Tal subconjunto pode ser obtido a partir da função

$$finais: U \rightarrow U,$$
$$finais(U) = \{u_i \mid u_i \in U \wedge pos(u_i) = \emptyset\}.$$

De modo geral, o conjunto de definições apresentados anteriormente tem o objetivo de formalizar os conceitos atinentes ao modelo de estruturação. Assim, a partir dessas definições visa-se a construção de um conjunto de modelos (e.g., notação gráfica e um modelo computacional) para a estruturação de *curricula* do *Mathema*.

4.2.2. Notação Textual

Visando facilitar a construção da estrutura de *curriculum* foi definida uma notação textual baseada em uma gramática livre de contexto. Com isso busca-se, aproximar a proposta de um modelo computacional, haja vista que tal linguagem pode ser operacionalizada por um compilador. Desse modo, a linguagem para especificar o modelo[7], tal como pode ser visto na

[7] Disponível em http://goo.gl/OXr7h.

Tabela 1, é composta basicamente por três partes, a saber: (1) declaração das unidades pedagógicas; (2) definição da taxonomia; e (3) definição das relações de ordem.

Assim, compete à primeira parte declarar todas as unidades pedagógicas que irão compor o *curriculum*, de modo que cada unidade recebe uma identificação única e um nome ou rótulo que determina o tópico dessa unidade. A segunda parte compete definir a taxonomia das unidades pedagógicas e, desse modo, visa construir a árvore de unidades. Por fim, a terceira e última parte compete definir as relações de ordem (i.e., relações de obrigatoriedade, alternância ou opcionalidade) entre as unidades pedagógicas. Assim, diante da linguagem exposta, o exemplo a seguir tem o objetivo de demonstrar a estruturação de um *curriculum* de operações aritméticas básicas em frações numéricas. Este exemplo toma como base a mesma estrutura apresentada anteriormente no Exemplo 2.

Tabela 1: Linguagem de estruturação do *curriculum*

#Definição das estruturas que compõem a criação de um curriculum	
estrutura	::= (declaracao)+ (taxonomia)* (ordenacao)*
#Declaração das unidades pedagógicas que irão compor o curriculum	
declaracao	::= UP “=” “″” NOME “″” “;”
# Definição da taxonomia das unidades pedagógicas	
taxonomia	::= UP “=” “(” lista “)” “;”
# Definição das relações de ordem das unidades pedagógicas	
ordenacao	::= UP “=” “obg” “(” lista “)” “;” | UP “=” “opc” “(” lista “)” “;” | UP “=” “alt” “(” UP “,” lista “)” “;”
# Definição de símbolos auxiliares	
lista	::= UP (“,” UP)*
UP	::= “up” NUM
NUM	::= “0” | “1” .. “9” (“1” .. “9”)*
NOME	::= “a” .. “z” (“a” .. “z” | “_” | “ ” | NUM)

Exemplo 3: Notação Textual do *Curriculum* de Fração

Dado o domínio frações numéricas, um recorte desse domínio pode ser descrito com base na linguagem de estruturação definida anteriormente. Desse modo, primeiramente é necessário declarar as unidades pedagógicas que irão compor o curso (vide

Tabela 2).

Tabela 2: Declaração dos tópicos de fração

$up0 = \text{"Frações numéricas"};$
$up1 = \text{"Conceituação"};$
$up2 = \text{"Operações"};$
$up3 = \text{"Definição"};$
$up4 = \text{"Tipologia"};$
$up5 = \text{"Número Misto"};$
$up6 = \text{"Equivalência"};$
$up7 = \text{"Propriedades"};$
$up8 = \text{"Unárias"};$
$up9 = \text{"Binárias"};$
$up10 = \text{"Simplificação"};$
$up11 = \text{"Redução"};$
$up12 = \text{"Transformação"};$
$up13 = \text{"Comparação"};$
$up14 = \text{"Adição"};$
$up15 = \text{"Subtração"};$
$up16 = \text{"Divisão"};$
$up17 = \text{"Multiplicação"};$

Dando continuidade a estruturação do curso, é necessário explicitar as relações taxonômicas entre as unidades, ou seja, definir uma organização hierárquica entre os diversos tópicos que serão abordados no decorrer do curso. Assim, como pode ser visto na Tabela 3, a

primeira atribuição define os subtópicos da unidade $up0$ e, analogamente, as demais atribuições definem o restante da taxonomia.

Tabela 3: Taxonomia dos tópicos de fração

$up0 = (up1, up2);$
$up1 = (up3, up4, up5, up6, up7);$
$up2 = (up8, up9);$
$up8 = (up10, up11, up12);$
$up9 = (up13, up14, up15, up16, up17);$

Por fim, devem ser definidas as relações de ordem entre os diversos tópicos da estrutura. Nesse contexto, é importa frisar novamente que tais relações somente podem ser definidas entre tópicos de um mesmo nível, ou seja, entre tópicos dito irmãos. Assim, como pode ser visto na Tabela 4, as declarações definem relações de obrigatoriedade desde o primeiro até o último tópico do curso, ou seja, cria-se um fluxo que obriga a exploração de todas as unidades em ordem.

Tabela 4: Relações de ordem dos tópicos de fração

$up1 = obg(up2);$
$up3 = obg(up4);$
$up4 = obg(up5);$
$up5 = obg(up6);$
$up6 = obg(up7);$
$up8 = obg(up9);$
$up10 = obg(up11);$
$up11 = obg(up12);$
$up13 = obg(up14);$
$up14 = obg(up15);$
$up15 = obg(up16);$
$up16 = obg(up17);$

Finalizando, a notação textual aqui definida tem como objetivo prover uma linguagem que possa ser automaticamente processada por entidades de *software*. Nessa perspectiva, tal linguagem também poderia ser utilizada pelos construtores de *Mathema* na estruturação do *curriculum*, todavia frisa-se que tal linguagem não representa uma solução intuitiva para, por exemplo, professores. Assim, adiante serão apresentados outros mecanismos que visam solucionar tal problema.

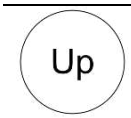
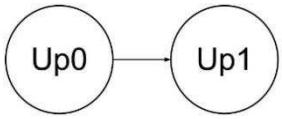
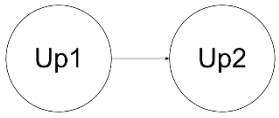
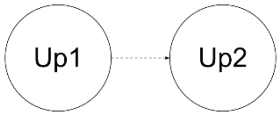
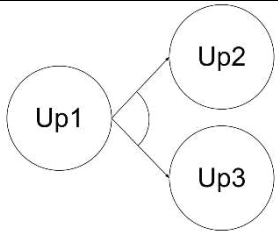
4.2.3. Notação Gráfica

Com base na linguagem definida anteriormente, foi construída uma notação gráfica que tem o objetivo de prover uma forma de abstração mais intuitiva e, consequentemente, facilitar a estruturação do *curriculum*. Essa notação visa definir uma linguagem visual que, futuramente, poderá ser acoplada a alguma ferramenta de autoria. Assim, como pode ser visto na Tabela 5, a linguagem gráfica define uma unidade pedagógica como um círculo com uma identificação inscrita ao mesmo. Os relacionamentos são representados por setas, de modo que o fluxo do relacionamento é representado pelo sentido da seta. Além disso, são previstos dois planos distintos: (1) o plano taxonômico; e (2) o plano de ordenação.

O plano taxonômico tem o objetivo de organizar a hierarquia da estrutura, tal como é feito nos livros didáticos. Assim, no que tange esse plano, fica definido que uma seta com linha cheia representa uma relação de uma unidade pai para com uma unidade filha. Desse modo, fixa-se o nó de origem como o tópico mais geral e, consequentemente, o nó de destino como o tópico mais específico. Com essa notação se define uma estrutura similar a uma árvore e, por conseguinte, não devem ser construídos ciclos na estrutura e tampouco deve haver desconectividade entre os nós.

O plano de ordenação tem o objetivo de definir a ordem que os tópicos devem ser explorados, definindo o fluxo de execução da estrutura de *curriculum*. Nesse sentido, no que diz respeito ao plano de ordenação, ficam definidos três tipos de relacionamentos: (1) obrigatório; (2) opcional; e (3) alternativo.

Tabela 5: Notação gráfica da estrutura de *curriculum*

Notação Geral	**Descrição**
	Unidade pedagógica
	Relação
Plano Taxonômico	
	Relação de taxonomia entre $up0$ e $up1$
Plano de Ordenação	
	Relação de obrigatoriedade de $up1$ para $up2$
	Relação de opcionalidade de $up1$ para $up2$
	Relação de alternância de $up1$ para $up2$ e $up3$

- **Obrigatório**: representado por uma seta com linha cheia, define uma relação de obrigatoriedade do nó origem para com o nó destino, ou seja, o nó destino deve ser obrigatoriamente explorado após o nó de origem e o nó de origem não pode ser explorado antes do nó de destino;

- **Opcional**: representado por uma linha tracejada, define uma relação de opcionalidade do nó origem para com o nó destino, ou seja, o nó destino pode ser explorado após o nó de origem e o nó de origem não pode ser explorado antes do nó de destino;

- **Alternativo**: representado por duas setas cheias e um arco, define uma relação de alternância entre o nó origem e os nós de destino. Esse relacionamento define que pelo menos um dos nós de destino deve ser explorado após o nó de origem e que nenhum nó de destino pode ser explorado antes do nó de origem.

Exemplo 4: Notação Gráfica do *Curriculum* de Fração

Dada a mesma estrutura apresentada no exemplo anterior, sua notação gráfica pode ser representada tanto no plano taxonômico quanto no plano de ordenação. Assim, na Figura 9Erro! Fonte de referência não encontrada. pode ser visto o plano taxonômico, com a estrutura hierárquica dos tópicos do curso. Em contrapartida, na Figura 10 pode ser visto o plano de ordenação, onde estão definidas as relações de ordem entre as unidades irmãs.

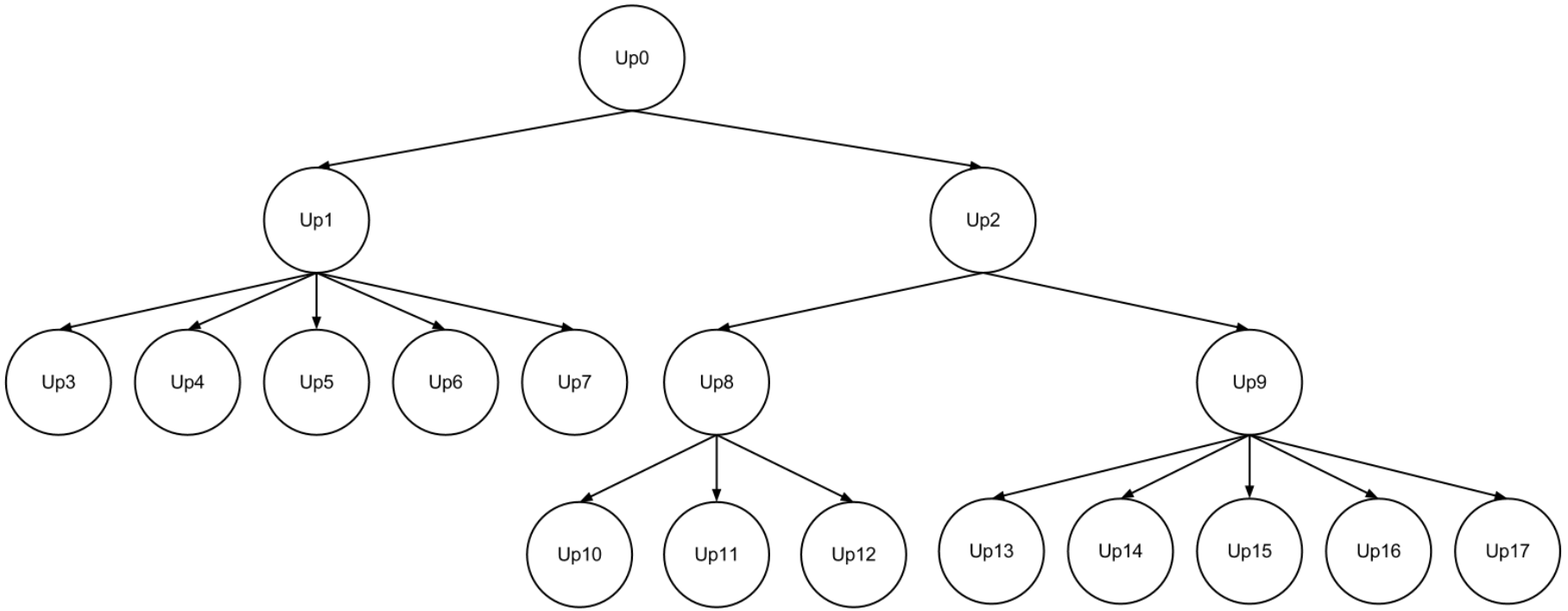


Figura 9: Taxonomia do *curriculum* de fração

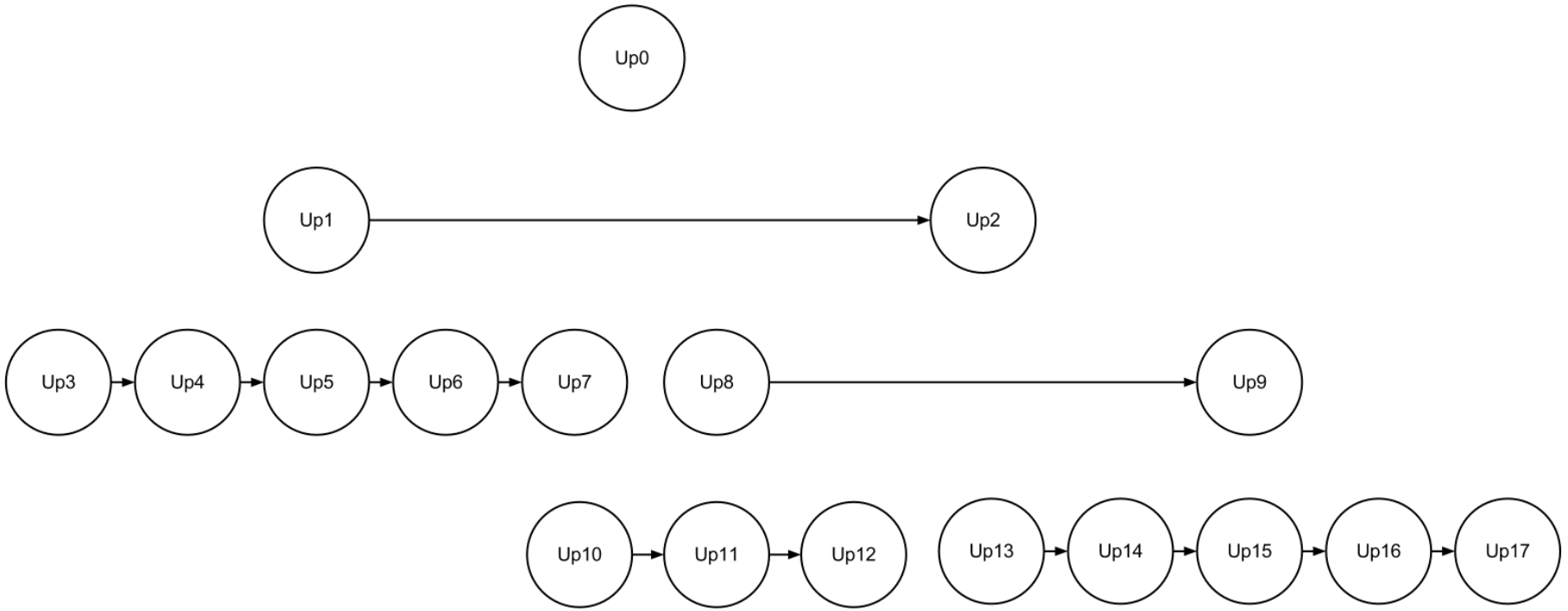


Figura 10: Ordenamento do *curriculum* de fração

4.2.4. Modelo Computacional

O modelo computacional foi construído para que entidades de *software* possam automaticamente manipular e extrair informações da estrutura curricular. Para a sua construção foi utilizada uma ontologia[8] baseada utilizando Lógica de Descrição [27]. Essa linguagem provê suporte para, por exemplo, restrições de cardinalidade, quantificador existencial, quantificador universal, conjunções, restrições, negações, entre outros. Escolheu-se investir no uso de ontologias, primeiramente, por utilizar uma linguagem forma, pelo seu alto poder de expressividade, pela possibilidade de verificação de consistência lógica e pela sua facilidade de compartilhamento, haja vista o advento da *Web* Semântica. Assim, como pode ser visto na Tabela 6, o modelo computacional tomou como base o formalismo definido na Seção 4.2.1.

Tabela 6: Ontologia do modelo de estruturação

# Definição do curriculum	
$Curriculum$	$\sqsubseteq Thing$
$Curriculum$	$\sqsubseteq = 1\ hasUnit\ PedagogicalUnit$
#Definição da unidade pedagógica	
$PedagogicalUnit$	$\sqsubseteq Thing$
$PedagogicalUnit$	$\sqsubseteq = 1\ hasState\ State$
$PedagogicalUnit$	$\sqsubseteq \forall\ hasChild\ PedagogicalUnit$
PedagogicalUnit	$\sqsubseteq \forall\ hasSequence\ UnitSequence$
# Definição do sequenciamento entre as unidades	
$UnitSequence$	$\sqsubseteq Thing$
$UnitSequence$	$\sqsubseteq = 1\ hasOperator\ Operator$
$UnitSequence$	$\sqsubseteq \exists\ hasSource\ PedagogicalUnit$
$Operator$	$\sqsubseteq Thing$
$Operator$	$\equiv \{Alternative\} \sqcup \{Mandatory\} \sqcup \{Optional\}$

[8] Disponível em http://goo.gl/OXr7h.

Sob essa perspectiva, tomando como base a estruturação do *curriculum* é possível definir um algoritmo de exploração dessa estrutura, levando-se em consideração a taxonomia, as relações de ordem e os estados de cada um dos nós. Desse modo, como mostrado no Algoritmo 1, uma função para a exploração da estrutura curricular recebe como entrada o grafo de conceitos e o nó raiz. Em suma, a exploração dos nós ocorre, inicialmente, do nó raiz até os nós de maior profundidade, para isso são utilizados os relacionamentos taxonômicos. Além disso, a cada nível é feita uma escolha dentre os nós iniciais, ou seja, os nós que não apresentam pré-requisitos. A função escolha pode ser vista como um ponto de variabilidade do algoritmo e, desse modo, pode implementar diferentes estratégias como, por exemplo, escolha aleatória, escolha tomando com base no modelo do estudante, entre outros. Assim, uma vez escolhido o nó inicial, é utilizada a função de escolha para definir a ordem de exploração dos nós posteriores.

Algoritmo 1: Algoritmo de exploração da estrutura de *curriculum*

Função: $explore(G, u_0)$
Entrada: Grafo G e nó inicial u_0
Saída: Grafo G com estados atualizados

// Explora os filhos iniciais do nó u_0
para cada $u \leftarrow escolha\left(iniciais\left(filhos(u_0)\right)\right)$ **faça**
    se $estado(u) = 0$ **então**
        $G \leftarrow explore(G, u)$;
    fim
fim

// Verifica se o nó u_0 *possui pré-requisitos pendentes*
para cada $u \leftarrow pre(u_0)$**faça**
    se $estado(u) = 0$ **então**
        retorna G;
    fim
fim

$estado(u) \leftarrow 1$; *Atualiza estado do nó* u_0

\\ *Explora os nós disponíveis*
para cada $u \leftarrow escolha\left(pos(u_0)\right)$ **faça**
    se $estado(u) = 0$ **então**
        $G \leftarrow explore(G, u)$;
    fim
fim

retorna G; \\ *Retorna a estrutura com os estados atualizados*

4.3. Modelo de Verificação

O modelo de verificação tem o objetivo de verificar propriedades na estrutura de *curriculum* construída pelo especialista. Para a definição desse modelo foi utilizada uma rede de Petri [28]. A rede é construída automaticamente com base na estrutura de *curriculum*. Para tal, foram definidos um conjunto de equivalências entre o modelo de *curriculum* e o modelo de verificação aqui proposto.

4.3.1. Definição do Modelo

Uma rede de Petri é um tipo particular de grafo direcionado que possui um estado inicial chamado de marca inicial. O grafo RP da rede de Petri é direcionado, ponderado e bipartite, consistindo de dois tipos de nós, os lugares e as transições. Os arcos interligam um lugar a uma transição ou vice-versa e, além disso, possuem um peso associado.

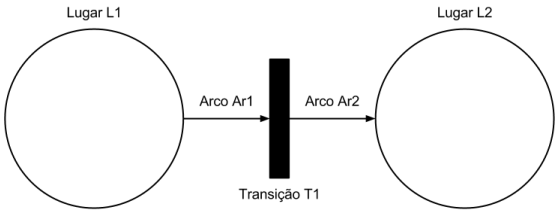


Figura 11: Rede de Petri e seus elementos básicos

Graficamente os lugares são representados por círculos e as transições por caixas, como mostrado na Figura 11. Arcos são rotulados com pesos (inteiros positivos) e pesos unitários geralmente são omitidos. Além disso, um número não negativo de fichas pode ser atribuído para cada lugar. Formalmente, uma rede de Petri pode ser definida tal como apresentada na Definição 10.

Definição 10: Seja RP uma rede de *Petri*, um enfoque para essa rede pode ser definida por $RP = (L, Tr, Ar, P, M_0)$. Assim, tem-se que:

i) A RP associa-se um conjunto L = $\{l_1, \cdots, l_i, \cdots, l_{|L|}\}$, onde cada $l_i \in L$ representa o i-ésimo lugar da rede;

ii) A RP associa-se um conjunto $Tr = \{tr_1, \cdots, tr_j, \cdots, tr_{|Tr|}\}$, onde cada $tr_j \in Tr$ representa a j-ésimo transição da rede;

iii) A RP associa-se um conjunto $Ar = \{\, ar_k \mid ar_k \in (L \times Tr) \cup (Tr \times L) \,\}$, onde $ar_k \in Ar$ representa um arco que liga um lugar a uma transição ou vice-versa;

iv) A RP associa-se uma função P: $Ar \rightarrow \mathcal{N}^*$, que tem o objetivo de ponderar os arcos;

v) A RP associa-se uma marca inicial M_0: $L \rightarrow \mathcal{N}^*$, que representa o estado da rede.

Um modelo de redes de *Petri* descreve os estados e eventos do sistema e pode ser utilizado para realizar simulações a fim de investigar diferentes cenários e explorar o comportamento do sistema. Com isso é possível observar os efeitos de cada passo da execução do modelo. Para simular o comportamento do sistema, os estados ou fichas da rede de Petri são alterados de acordo com as funções definidas na Definição 11.

Definição 11: Seja RP uma rede de Petri, um conjunto de funções que tornam possível simular a dinâmica do sistema pode ser dado por:

i) Dado um nó $n_i \in (L \cup Tr)$, existe um conjunto de nós anteriores ao mesmo que pode ser obtido a partir da função

$preSet: (L \cup Tr) \rightarrow (L \cup Tr)$,
$preSet(n_i) = \{\, n_j \mid (n_j \times n_i) \subseteq Ar \,\}$;

ii) Dado um nó $n_i \in (L \cup Tr)$, existe um conjunto de nós posteriores ao mesmo que pode ser obtido a partir da função

$postSet: (L \cup Tr) \rightarrow (L \cup Tr)$,
$postSet(\mathrm{n_i}) = \{n_k | (\mathrm{n_i} \times \mathrm{n_k}) \subseteq \mathrm{Arc}\}$;

iii) Dado uma transição $tr \in Tr$, a função que define se a transição está habilitada pode ser formalizada a partir da função

$habilitada: M \times Tr \rightarrow \{0, 1\}$,
$habilitada(M_i, tr_j) = \{1 \mid tr \in Tr \wedge (\forall l \in preSet(tr_j): M_i(l) \geq P((l, tr_j)))\}$ *e*
$habilitada(M_i, tr_j) = 0, nos\ demais\ casos.$

Além disso, a dinâmica de uma rede de Petri pode ser definida com base nos itens a seguir:

iv) Uma transição $tr \in Tr$ está habilitada se cada lugar de entrada $l \in L$ de tr está marcada com pelo menos $P(l, tr)$ fichas, onde $P(l, tr)$ é o peso do arco que vai do lugar l para a transição tr;

v) O disparo de uma transição tr remove $P(l, tr)$ fichas de cada lugar de entrada l de tr, e adiciona $P(tr, l)$ fichas para cada lugar de saída l de tr.

O comportamento da reação química $2H_2 + O_2 \rightarrow 2H_2O$ pode, por exemplo, ser modelada utilizando as definições apresentadas anteriormente. Neste exemplo, tal como apresentado na Figura 12, existem primeiramente duas fichas para cada lugar de entrada, habilitando a transição. Após o disparo da transição, cada lugar de entrada tem um número de fichas consumido de acordo com o peso de seu arco correspondente e, por conseguinte, o lugar de destino tem um número de fichas produzidas de acordo com o peso de seu arco de origem.

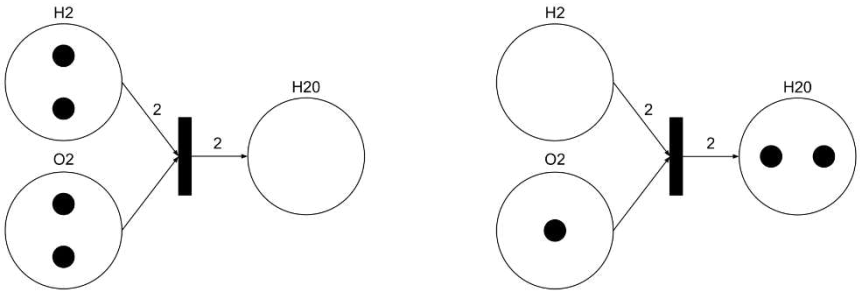


Figura 12: Dinâmica de uma rede de Petri

4.3.2. Modelo de Transformação

Uma vez apresentadas as definições do modelo de verificação, é preciso construir um algoritmo que possibilite realizar a transformação do modelo de estruturação do *curriculum* (vide seção 4.2) para o modelo de verificação. Nesse sentido, visa-se construir uma rede de Petri equivalente à estrutura curricular definida pelo especialista e, a partir dessa rede, verificar o comportamento da estrutura, ou seja, realizar simulações da dinâmica de interação da estrutura de *curriculum*. Assim, em virtude do discutido, a Definição 12 apresenta uma função para a construção de estruturas em rede de Petri que sejam equivalentes as relações de ordem propostas no modelo de estruturação de *curriculum.*

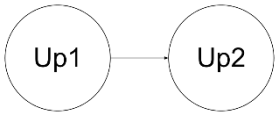
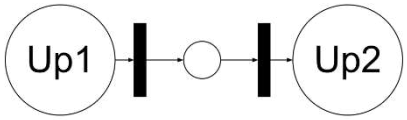
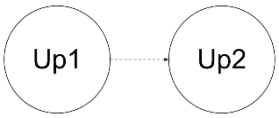
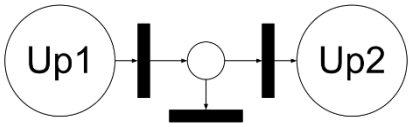
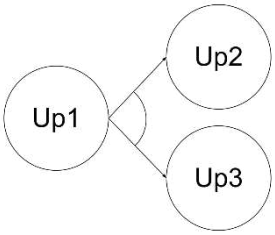
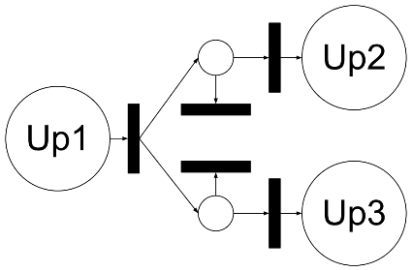
Definição 12: Seja $(u_i, u_j) \in O$ uma relação de ordem entre dois nós $u_i, u_j \in U$, existe uma função que constrói uma rede de Petri resultante que simula o comportamento dessa estrutura, tal função pode ser formalizada tal como segue

$$criarEstrutura: U \times U \times O \rightarrow RP,$$
$$criarEstrutura(u_i, u_j, o_k) = \{rp \mid u_i \in U \wedge u_j \in U \wedge o_k \in O \wedge rp \in RP\}.$$

O comportamento dessa função está descrito na

Tabela 7, que apresenta as estruturas curriculares e suas redes de Petri equivalentes. Nesse sentido, essas redes de Petri apresentam uma dinâmica de interação equivalente ao comportamento esperado pela execução da estrutura curricular. Assim, essas estruturas podem ser vistas como subestruturas de uma rede de Petri correspondente a toda estrutura de *curriculum*. Sob essa perspectiva, a função de criação dessas estruturas será utilizada em conjunto com um algoritmo que visa construir uma rede de Petri equivalente à estrutura de *curriculum* definida pelo especialista.

Tabela 7: Estruturas curriculares e redes de Petri resultantes

Curriculum	**Rede de Petri**
	
	
	

Dando continuidade ao modelo de transformação, o Algoritmo 2 juntamente com o Algoritmo 3 apresentam os passos para a transformação da estrutura de *curriculum* em uma rede de Petri. Ao primeiro algoritmo compete explorar a estrutura curricular desde o nó mais geral (a.k.a., nó raiz) até os nós mais específicos (a.k.a., nós folha). Tal exploração leva em consideração as relações de ordem definidas na estrutura e com base nessas relações constrói recursivamente cada subestrutura da rede de Petri. Ao segundo algoritmo compete fazer a ligação entre um nó e seus posteriores. Tal ligação deve levar em consideração o tipo de relação de ordem, bem como critérios de taxonomia, haja vista que um nó deve estar sempre ligado aos primeiros descendentes (i.e., aos nós folha iniciais) dos seus nós posteriores.

Algoritmo 2: Algoritmo de construção da rede de Petri

Função: $criarRP(u_0,\ conjuntoUps)$
Entrada: Nó inicial u_0 do grafo e conjunto vazio para identificar os nós transformados
Saída: Rede de Petri *RP* resultante da transformação

RP // Declaração da estrutura da rede de Petri

// Se o nó em foco já tiver sido transposto para a rede de Petri
se $u_0 \in conjuntoUps$ **então**

    retorna RP;
fim

// Explora o grafo em profundidade selecionando apenas os nós finais
para cada $u \leftarrow finais(filhos(u_0))$ **faça**

    // Adiciona na rede os componentes de uma sub-rede correspondente ao nó filho
    $RP.addAll(criarRP(u, conjuntoUPS))$;
fim

// Explora os pré-requisitos do nó em foco
para cada $u \leftarrow pre(u_0)$ **faça**

    // Adiciona na rede os componentes de uma sub-rede correspondente ao nó anterior
    $RP.addAll(criarRP(u, conjuntoUPS))$;
fim

// Explora os nós posteriores ao nó em foco
para cada $u \leftarrow pos(u_0)$ **faça**

    // Adiciona na rede os componentes de uma sub-rede correspondente ao nó posterior
    $RP.addAll(criarJunção(u_0, u))$;
fim

se $pos(u_0) = \emptyset\ AND\ pai(u_0) \neq \emptyset$ **então**

    // Adiciona na rede uma estrutura obrigatória entre filho e pai
    $RP.addAll(criarEstrutura(u_0, pai(u_0), Obg))$;
fim

// Adiciona o nó ao conjunto como forma de identificar a finalização do mesmo
$conjuntoUps.add(u_0)$;

retorna RP; \\ *Retorna a rede de Petri resultante das transformações*

Algoritmo 3: Algoritmo de construção dos nós posteriores

Função: $criarJunção(u_0, u_1)$
Entrada: Nó inicial u_0 e nó posterior u_1
Saída: Rede de Petri *RP* resultante da transformação

// Se o nó posterior não tiver filhos
se $filhos(u_1) = \emptyset$ **então**

    // Adiciona na rede uma estrutura correspondente à relação dos nós
    retorna $criarEstrutura(u_0, u_1, relacao(u_0, u_1))$;
fim

RP // Declaração da estrutura da rede de Petri

// Explora os filhos inicias do nó em foco
para cada $u \leftarrow iniciais(filhos(u_1))$ **faça**

    // Adiciona na rede os componentes de uma sub-rede correspondente aos filhos
    $RP.addAll(criarJunção(u_0, u))$;
fim

retorna *RP*; \\ *Retorna a rede de Petri resultante das transformações*

Por fim, no intuito de tornar mais claro o funcionamento do algoritmo, o exemplo a seguir apresenta a construção da rede de Petri correspondente à estrutura curricular do domínio de fração.

Exemplo 5: Rede de Petri do *Curriculum* de Fração

Dada a estrutura curricular de fração apresentada nos exemplos anteriores, a aplicação do algoritmo irá construir uma estrutura que apresente uma dinâmica equivalente ao *curriculum* definido. Desse modo, como apresentado na Figura 13, a rede de Petri correspondente ao grafo do *curriculum* de fração apresenta um fluxo de execução convencional. Nesse caso em particular o fluxo de execução foi equivalente ao de um caminhamento de árvore em pós ordem, todavia é importante ressaltar que essa não é uma regra para a construção da rede. Nesse sentido, frisa-se que outras construções curriculares poderão apresentar comportamentos não convencionais e, consequentemente, a construção da rede de Petri não será uma tarefa trivial. Assim, destaca-se novamente a importância do modelo de transformação, que irá construir a rede de Petri automaticamente.

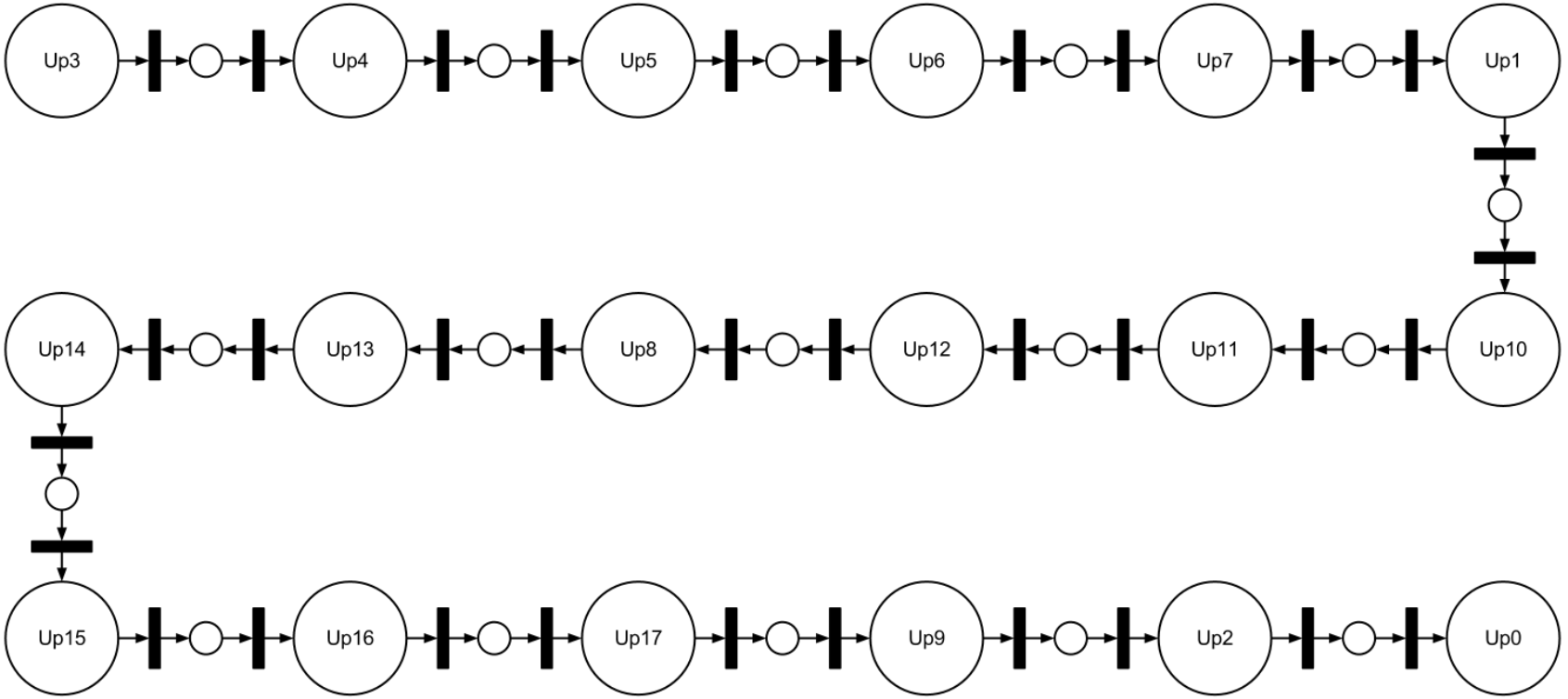


Figura 13: Rede de Petri do *curriculum* de fração

4.3.3. Modelo Computacional

Para o modelo computacional investiu-se em uma abordagem baseada em redes de Petri coloridas [26]. Esse tipo de rede foi escolhido por apresentar um nível de expressividade em conformidade com o do modelo de estruturação de *curriculum* definido nessa dissertação. Além disso, redes de Petri coloridas são amplamente difundidas tanto na esfera acadêmica quanto na esfera industrial. Por fim, esse tipo de rede tem disponível a ferramenta CPN *Tools*[9], que é uma solução consolidada para redes de Petri coloridas e que está em constante evolução.

Nesse contexto, para a construção do modelo computacional pode-se investir no arcabouço *Access/CPN* [29], que provê um conjunto de classes em Java para a manipulação de modelos de redes de Petri da ferramenta CPN *Tools*. Assim, é possível implementar o Algoritmo 2 e o Algoritmo 3, para a transformação do modelo de *curriculum*. Uma vez construída a rede de Petri como, por exemplo, a apresentada na Figura 14, é possível realizar simulações. Na simulação apresentada é possível notar o estado de quatro alunos. Os alunos 1 e 3 encontram-se na unidade de número um, o estudante 4 encontra-se na unidade de número dez e, por fim, o estudante 2 encontra-se na unidade de número oito.

[9] Mais informações em http://cpntools.org/.

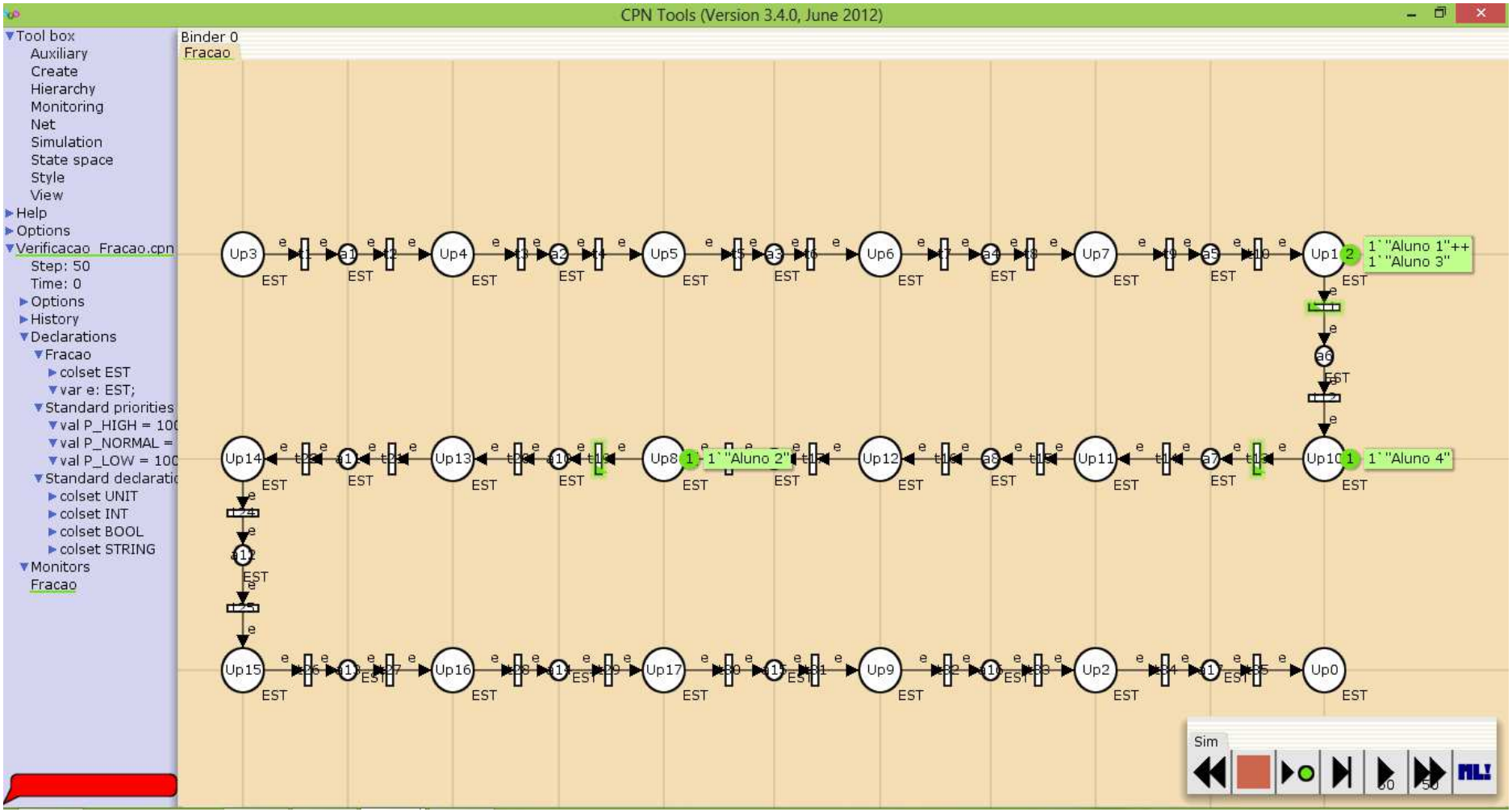


Figura 14: Rede de Petri do *curriculum* de fração no CPN *Tools*

4.4. Modelo de Agentes

O modelo de agentes tem a função de definir a estrutura dos agentes que irão compor SATA. Nesse contexto, tal modelo provém, além de um conjunto de definições, um algoritmo de transformação da estrutura de *curriculum* para a sociedade de agentes. Além disso, tal modelo provê uma revisão da arquitetura de um agente *Mathema*, adequando sua estrutura a conceitos mais atuais, bem como uma nova estruturação para SATA.

4.4.1. Definição do Modelo

Para a definição de um agente será utilizada a visão apresentada por Russel & Norvig [30], onde um agente é definido como uma entidade autônoma capaz de perceber um ambiente e agir sobre o mesmo tomando como bases essas percepções. Desse modo, um agente, tal como pode ser visto na Figura 15 apresenta basicamente dois mecanismos: (1) sensores; e (2) atuadores. Os sensores são quaisquer mecanismos (e.g., câmera, microfone, termostato, sistema de troca de mensagens etc.) que obtém informações do ambiente. A partir desses sensores é que um agente poderá captar mudanças no ambiente. Os atuadores são quaisquer mecanismos (e.g., sistema de som, braço mecânico, monitor de vídeo, impressora etc.) que atuam no ambiente algum modo, produzindo algum novo recurso ou alterando os recursos existentes no mesmo. Sob essa perspectiva as definições que seguem tem o objetivo de formalizar o conceito de agentes, bem como definir as entidades que irão compor SATA.

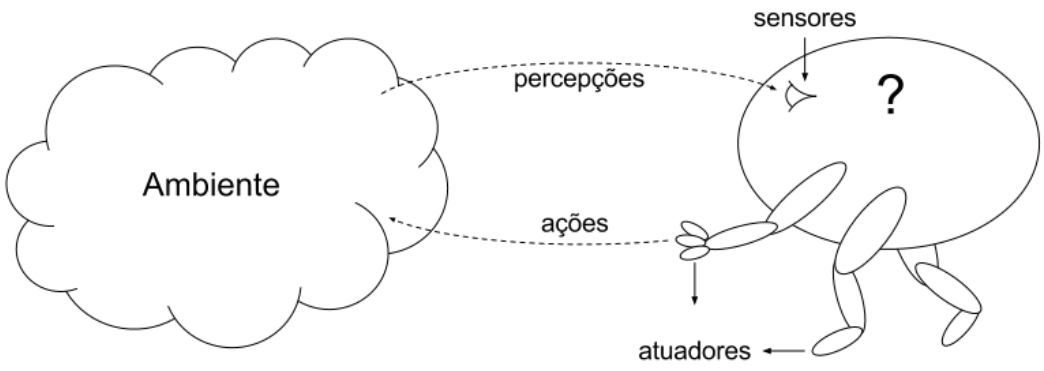


Figura 15: Modelo conceitual de um agente
Fonte: Adaptado de [30]

Definição 13: Seja A um agente sobre o qual se pretende construir uma especificação, um enfoque para esse agente pode ser definido por $A = (S, Pe, At)$. Assim, tem-se que:

i) A A associa-se um conjunto $S = \{s_1, \cdots, s_i, \cdots, s_{|S|}\}$, onde cada s_i representa um sensor particular do agente em foco;

ii) A A associa-se um conjunto $Pe = \{pe_1, \cdots, pe_j, \cdots, pe_{|Pe|}\}$, onde cada p_j representa uma percepção particular que o agente obtém do ambiente;

iii) A A associa-se um conjunto $At = \{at_1, \cdots, at_k, \cdots, at_{|At|}\}$, onde cada at_k representa um atuador particular do agente em foco.

Uma vez definida a estrutura de um agente é preciso formalizar o ciclo comportamental do mesmo. Este ciclo, tal como apresentado na Definição 14, define o comportamento de um agente, ou seja, como o mesmo pode interagir com o ambiente.

Definição 14: Seja A um agente em foco, seu ciclo comportamental pode ser representado por um conjunto de funções, tal como segue:

i) Seja S o conjunto de sensores do agente A e Pe seu conjunto de percepções, um mecanismo que mapeia os sensores desse agente em suas percepções é definido pela função

$$perceber : S \rightarrow Pe,$$
$$perceber(s_i) = \{pe_j \mid s_i \in S \wedge pe_j \in Pe\};$$

ii) Seja Pe o conjunto de percepções de um agente A e At o conjunto de atuadores que este agente possui, um mecanismo que mapeia as percepções deste agente em ações do mesmo pode ser definido pela função

$$atuar : Pe \rightarrow At,$$
$$atuar(\mathrm{p_j}) = \{\mathrm{at_k} \mid \mathrm{p_j} \in \mathrm{P} \wedge \mathrm{at_k} \in \mathrm{At}\};$$

iii) Seja S o conjunto de sensores e At um conjunto de atuadores de um agente A um mecanismo que descreve o ciclo comportamental desse agente pode ser definido pela composição

$$atuar \circ perceber : S \rightarrow At,$$
$$(atuar \circ perceber)(s_i) = \{ at_k \mid s_i \in S \wedge at_k \in At \}.$$

Dando continuidade, é possível utilizar as definições anteriores a fim de definir os agentes que irão compor SATA. Tal sociedade de agentes será composta por dois tipos de agentes:

- **Agente Tutor (AT)**: agente que irá efetivamente desempenhar papéis como resolução de problemas, avaliação de soluções, recomendação de recursos entre outras tarefas diversas;

- **Agente Gerenciador (AG)**: agente que tem o papel de gerenciar os ATs. Esse tipo de será bastante demandado em situações que necessitem da colaboração (e.g., uma resolução distribuída de problemas) entre diversos ATs.

Assim, a definição de agente apresentada anteriormente se adéqua a essa visão, sobretudo no que diz respeito aos ATs. Todavia, é necessário definir uma extensão dessa definição para abranger também os AGs, tal como pode ser visto na Definição 15.

Definição 15: Seja AG um agente gerenciador sobre o qual se pretende construir uma especificação, um enfoque para esse agente pode ser definido por $AG = (S, Pe, At, A)$. Assim, tem-se que:

i) A AG associa-se os mesmos conjuntos $S, Pe\ e\ At$ apresentados na Definição 13;

ii) A AG associa-se um conjunto $A = \{a_1, \cdots, a_l, \cdots, a_{|A|}\}$, onde cada a_l representa um agente sob a tutela do gerente em foco.

Assim, tal como apresentado, os AGs fazem referência direta a um conjunto de agentes que estão sob a sua tutela. Tais agentes podem ser tanto ATs quanto outros AGs, formando uma estrutura hierárquica similar a definida pelo especialista na estruturação do *curriculum*.

4.4.2. Modelo de Transformação

O modelo de transformação tem o objetivo de prover um mecanismo automático para o apontamento dos agentes que irão compor a SATA. Esse mecanismo servirá como guia para que o Engenheiro de *Software* possa efetivamente construir os agentes da plataforma, todavia compete ao engenheiro o papel de condensar ou estratificar a sociedade de agentes quando achar necessário. Tais decisões de projeto fogem ao escopo desse mecanismo e devem ser tomadas por agentes humanos. Com isso em mente, o Algoritmo 4 apresenta o mecanismo de apontamento dos agentes tomando como base a estrutura de *curriculum*.

Algoritmo 4: Algoritmo de construção da sociedade de agentes

Função: $criarSata(u_0)$
Entrada: Nó inicial u_0 do grafo G
Saída: Gerente da sociedade de agentes

// Se o nó u_0 for uma folha então cria um agente tutor
se $filhos(u_0) = \emptyset$ **então**

    // Retorna um agente tutor referente ao tópico u_0
    retorna $criarTutor(u_0)$;
fim

// Cria um agente gerenciador
$AG \leftarrow criarGerente(u_0)$;

// Explora os filhos do nó u_0
para cada $u \leftarrow filhos(u_0)$**faça**

    // Adiciona ao gerente um agente referente a cada subtópico de u_0
    $AG.add(criarSATA(u))$;
fim

retorna AG; \\ *Retorna o gerente*

O algoritmo de transformação utiliza ao plano taxonômico da estrutura de *curriculum* para o apontamento dos agentes. Desse modo, obtém-se como resultado uma estrutura de agentes igual estrutura de tópicos do curso, onde cada agente reflete um tópico do curso. Para os tópicos mais específicos (i.e., os nós folha da árvore de tópicos) são criados Agentes Tutores, que tem a função de prover efetivamente as funcionalidades de tutoria para o estudante. Para os tópicos mais gerais (i.e., os nós da árvore que possuem filhos) são criados Agentes Gerenciadores, que tem a função de gerenciar as atividades dos agentes sob a sua tutela. Nessa perspectiva, obtém-se o agente mais geral (i.e., o agente resultante do nó raiz da árvore) e, a partir deste, pode-se explorar a árvore de agentes até o agente mais específico, que nesse caso representa o AT.

4.4.3. Modelo Computacional

Como dito anteriormente, para a construção dos agentes, investiu-se em uma combinação da arquitetura de proposta no *Mathema* e a visão apresentada por Russel & Norvig [30]. Esse tipo de arquitetura visa abstrair os mecanismos de interação de um agente para um conjunto de sensores e atuadores. Desse modo, qualquer tipo de interação do agente com o ambiente (i.e., tanto interações com entidades ativas quanto passivas) será efetivada via sensores e atuadores. Desse modo, a antiga camada de comunicação proposta por de Barros Costa [1] terá a função de realizar qualquer tipo de interação (i.e., interação entre agentes humanos, agentes de *software*, ou entidades passivas do ambiente).

Sob essa perspectiva, a arquitetura de um agente fica definida, tal como apresentada na Figura 16, como uma entidade composto por três camadas: (1) Sistema de Distribuição; (2) Sistema Social; e (3) Sistema Tutor.

- **Sistema de Distribuição**: compete perceber eventos de interesse do agente e, com bases nessas percepções, repassar para o sistema social. Além disso, tal sistema tem a função de receber notificações do sistema social e alocar um conjunto de atuadores para efetivar uma ação no ambiente;

- **Sistema Social**: compete armazenar as habilidades dos agentes situados no ambiente, bem como suas próprias habilidades. Desse modo, dada uma percepção enviada pelo sistema de interação, cabe a esse sistema enviar tal percepção para o resolvedor de problemas (caso o agente possua habilidades para tratá-la) ou então repassar tal percepção para outro agente que possa resolvê-la (caso o agente não possua habilidades para tratá-la);

- **Sistema Tutor**: compete processar as percepções recebidas pelo sistema social e atualizar as informações internas do agente, quanto necessário, e mapear tal percepção e um conjunto de ações a serem enviadas para os atuadores do agente.

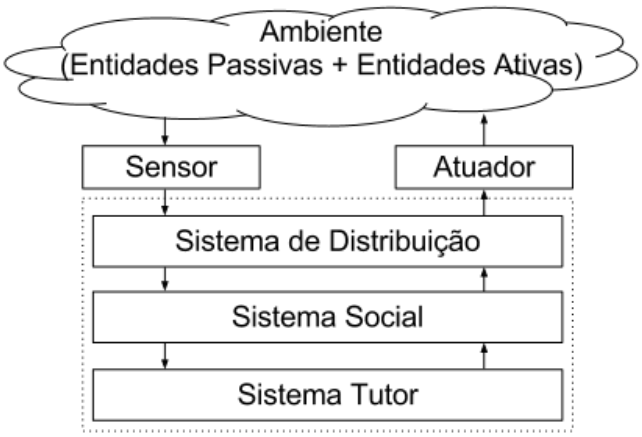


Figura 16: Visão macro de um agente *Mathema* – arquitetura revisitada

Em comparação com a arquitetura proposta por de Barros Costa [1], a presente arquitetura se mostra mais alinhada com uma visão de agentes que é amplamente difundida e aceita pela comunidade. Além disso, a arquitetura, tal como posta, concentra todas as interações do agente nas abstrações de sensor e atuador. Desse modo, é possível tratar da mesma forma diversos tipos de interação e, além disso, é possível adicionar mais facilmente novos tipos de interação não previstos, haja vista que esses mecanismos são tidos como pontos de extensão[10] na arquitetura. Outro ponto de destaque reside na arquitetura em camadas e no fluxo de execução das tarefas do agente. Na arquitetura proposta por de Barros Costa [1] existem dois pontos de entrada na arquitetura: (1) entrada pelo sistema de distribuição quando ocorre uma interação entre agentes; e (2) entrada pelo sistema tutor quando ocorre interação do estudante com o agente. Nesse sentido, a arquitetura revisada

[10] Do inglês *hot spot.*

define apenas um ponto de entrada via sensores e, consequentemente, pelo sistema de distribuição. Assim, a arquitetura se mostra mais concisa, pois define melhor os papéis de cada camada e abstrai os mecanismos de interação, tornando a arquitetura mais flexível.

Para fins de implementação, investiu-se em uma solução baseada no arcabouço Jade [24], para o desenvolvimento dos sistemas de distribuição e social. Além disso, para a construção do sistema tutor foi usado um *shell* de sistema especialista denominado *Inabit*[11], prestando-se a desenvolver cada um dos módulos de uma arquitetura clássica de um STI. Na definição da implementação foram inseridos diversos padrões de projeto [31] no intuito de prover uma solução de *software* mais flexível. Nesse sentido, os sensores de um Agente foram definidos com o padrão de projeto *Observer* e, desse modo, é possível adicionar e remover sensores mais facilmente. Analogamente, os atuadores foram definidos com o padrão *Command*, que tem o objetivo de flexibilizar a inserção e remoção de novos atuadores no Agente.

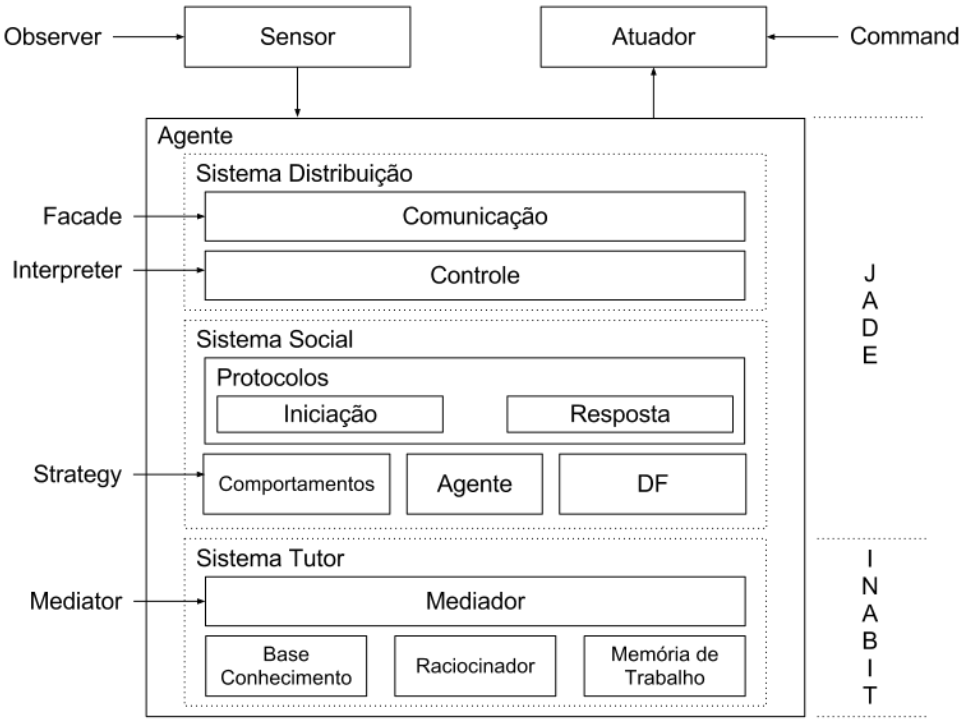


Figura 17: Visão micro de um agente *Mathema* – arquitetura revisitada

[11] Disponível em http://code.google.com/p/inabit/.

No que tange o sistema de distribuição foram definidos dois padrões de projeto: (1) padrão *Facade*, que define um ponto único de acesso às funcionalidades do Agente; e (2) padrão *Interpreter*, que tem o objetivo de interpretar as percepções do agente e decidir como tratá-las. Com relação ao sistema social são utilizados os conceitos definidos pelo arcabouço Jade (e.g., protocolos de interação, conceito de páginas amarelas e os comportamentos). Nessa camada é utilizado o padrão de projeto *Strategy*, que visa flexibilizar a mudança de comportamentos do Agente em tempo de execução. Além disso, novos comportamentos podem ser adicionados mais facilmente com esse padrão. Por fim, na última camada utilizou-se o padrão *Mediator*, que tem o objetivo de centralizar as funcionalidades em um único ponto de acesso e, além disso, mediar operações que necessitem utilizar múltiplos Raciocinadores. Assim, compete ao Mediador gerenciar todas as tarefas internas do agente, bem como o acesso à Base de Conhecimento e a Memória de Trabalho.

Dando continuidade ao modelo computacional, investiu-se em uma estruturação de agentes similar à estrutura de *curriculum* definida na seção 4.2. Nesse contexto, o modelo de classes conceitual, tal como apresentado na Figura 18, utiliza uma combinação de dois padrões de projeto:

- ***Composite*** é utilizado para estruturar as classes em um formato de árvore, tal como o definido pela estrutura de *curriculum*. Desse modo, um Agente Gerenciador pode ser composto por diversos agentes (i.e., tanto Agentes Tutores quanto outros Agentes Gerenciadores), formando assim uma estrutura taxonômica;

- ***Mediator*** é utilizado para estrutura as classes segundo uma estrutura que isole os agentes, fazendo com que cada agente conheça apenas o seu mediador. Desse modo, todas as interações entre agentes ocorrerão via um mediador e, assim, tarefas compostos podem ser efetuadas de modo transparente, haja vista que o agente mediador tem a função de gerenciar todo o processo.

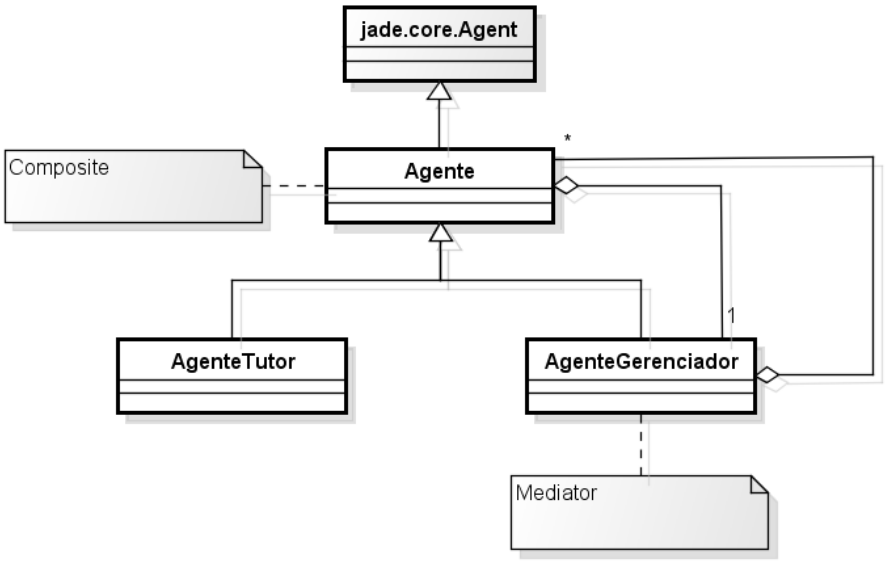


Figura 18: Modelo de classes conceitual de um agente *Mathema*

No intuito de tornar mais claro a estruturação dos agentes, o exemplo a seguir apresenta a construção dos agentes correspondente à estrutura curricular do domínio de fração.

Exemplo 6: Agentes do *Curriculum* de Fração

Dada a estrutura curricular de fração apresentada nos exemplos anteriores, a aplicação do algoritmo de construção de agentes irá apontar os agentes que farão parte de SATA. Assim, como apresentado na Figura 13, os agentes seguem uma estrutura igual a da taxonomia definida no *curriculum* de fração, com cada nó folha resultando em um AT e os demais nós resultando em AGs. Essa estruturação influência diretamente o funcionamento de SATA e sempre que um AG recebe uma requisição este a repassa para o agente mais adequado, que nesse caso pode ser tanto um dos seus subordinados ou o seu mediador.

Para exemplificar o funcionamento de uma funcionalidade de SATA a Figura 20 apresenta o processo de resolução da expressão $(1 / 2) + (2 / 3) \times (5 / 4)$. Primeiramente, o agente raiz de SATA recebe uma requisição contendo toda a expressão a ser resolvida e, de posso dessa expressão, a repassa para o agente mais adequado, que nesse caso é o de Operações. Por sua vez, o agente de Operações analisa a expressão e decide que por uma questão de precedência de operadores deve-se primeiramente resolver a subexpressão $(2 / 3) \times (5 / 4)$. Tal expressão compete ao agente de operações Binárias e, posteriormente, ao agente de Multiplicação que resolve a expressão. O resultado da expressão (i.e., o valor $(10 / 12)$) é

retornada para cada agente do encadeamento e cada agente analisa o resultado no intuito de verificar se alguma operação pode ser realizada. Sob essa perspectiva, o agente de Operações percebe que o resultado pode ser simplificado e, desse modo, faz uma requisição ao agente de operações Unárias e, consequentemente, ao agente de Simplificação. De posse do resultado simplificado o agente de Operações requisita ao agente de operações Binárias e, posteriormente, ao agente de Adição que realize a operação $(1 / 2) + (5 / 6)$ resultando no valor $(16 / 12)$. Como é possível notar, mais uma vez o resultado deve ser simplificado pelo agente de Simplificação resultando no valor $(4 / 3)$, contudo o agente de operações Unárias percebe que a fração é em foco é imprópria e requisita ao agente de Transformação que modifique o valor para uma fração mista $1\ (1 / 3)$. Por fim, a solução é repassada para os agentes da cadeia de resolução até retornar ao agente raiz.

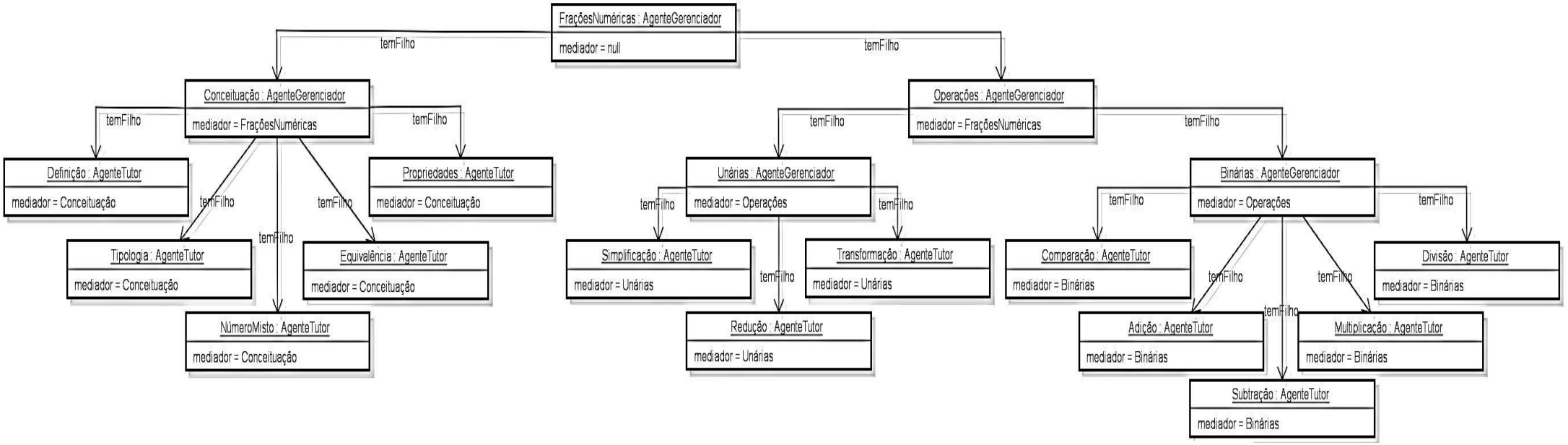


Figura 19: Diagrama de objetos dos agentes de fração

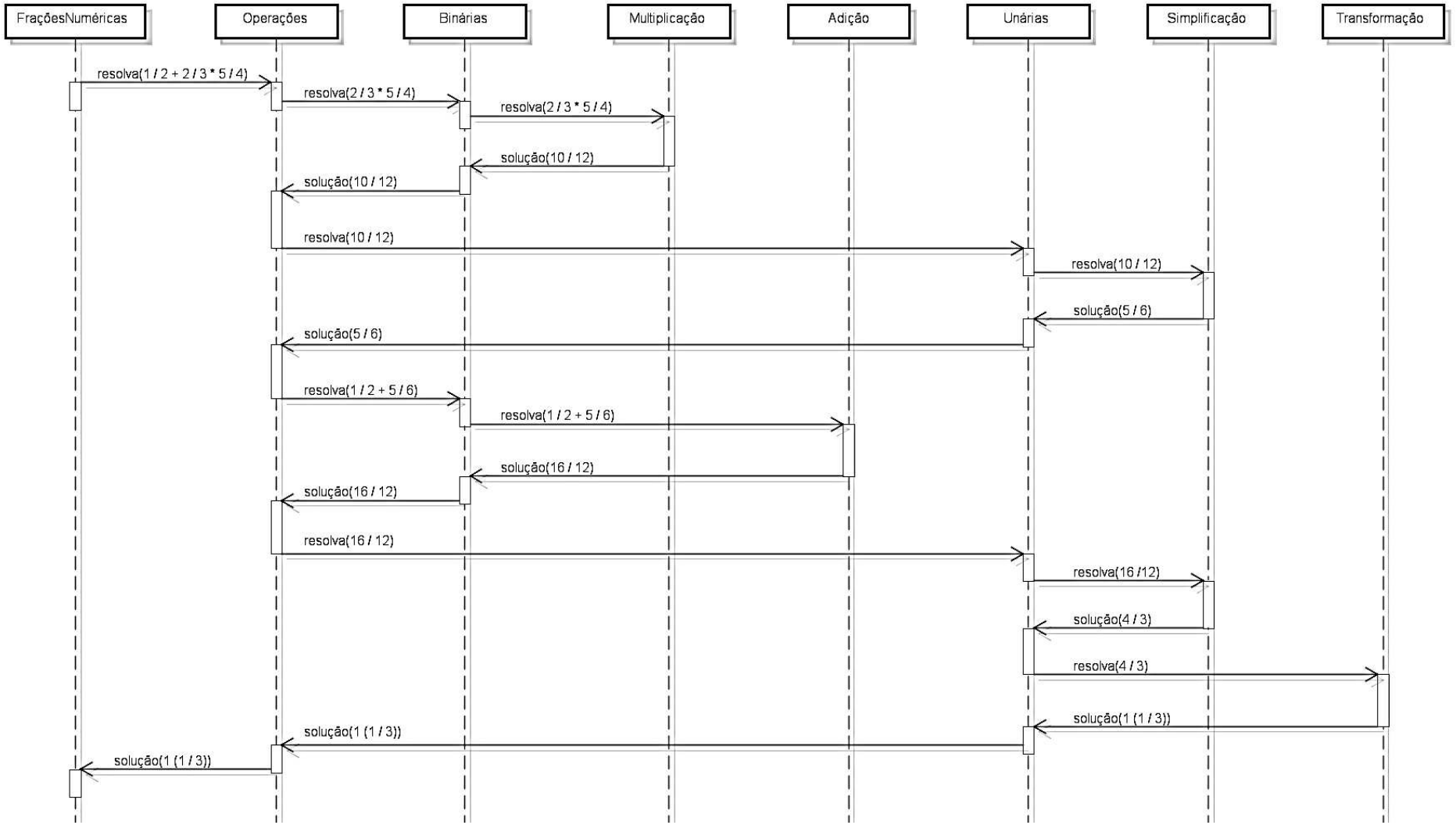


Figura 20: Diagrama de sequência resolução distribuída de problemas

5. AVALIAÇÃO

Para avaliar a viabilidade da abordagem proposta, o presente capítulo apresenta a utilização da sistemática e dos modelos construídos em três estudos de caso, a saber: (1) Curso de Ciência da Computação; (2) curso de Aprendizagem de Máquina; e (3) curso de Lógica Computacional. Obviamente, o presente estudo não tem o objetivo de cobrir os conteúdos dos cursos em sua totalidade, haja vista a diversidade de conhecimento e a profundidade de cada uma delas. Entretanto, ao longo dos casos serão ressaltadas algumas características de cada modelo.

5.1. Caso 1: Ciência da Computação

O presente caso investiu na estruturação do curso de Ciência da Computação da Universidade Federal de Alagoas. Desse modo, investiu-se na construção da grade curricular do curso[12] utilizando o modelo de estruturação do *curriculum*. Para essa estruturação investiu-se em duas abordagens distintas: (1) uma abordagem centrada na taxonomia; e (2) uma abordagem centrada nas relações de ordem. A primeira abordagem apresenta uma ênfase maior nas relações taxonômicas do curso, definindo uma ordem que segue estritamente os períodos do curso. A segunda abordagem abstrai os períodos do curso e enfatiza as relações de ordem, dando mais liberdade para o estudante.

5.1.1. Estruturação do *Curriculum*: uma abordagem centrada na taxonomia

Com base na estrutura proposta pelo programa do curso de Ciência da Computação da Universidade Federal de Alagoas, pode-se fixar esse domínio e construir uma visão curricular para o mesmo utilizando o modelo de estruturação. Assim, utilizando-se a linguagem para a estruturação do *curriculum* pode-se chegar à estrutura definida na Tabela 10. Essa estrutura é

[12] Disponível em http://www.ufal.edu.br/unidadeacademica/ic.

formada por uma unidade raiz, que representa o curso de Ciência da Computação em si, e por mais 53 unidades pedagógicas, que representam as matérias obrigatórias do curso.

Como dito anteriormente, essa abordagem enfatiza as relações taxonômicas do curso, respeitando as relações de ordem entre períodos do curso. Nessa perspectiva, foram definidas 8 unidades que visam apenas estruturar o curso segundo um agregado de períodos. Assim, como apresentado na Tabela 8, as unidades que vão desde $up1$ até $up8$ tem unicamente a função de isolar as matérias do curso segundo um conjunto de períodos.

Tabela 8: Taxonomia dos tópicos de Ciência da Computação

$up0 = (up1, up2, up3, up4, up5, up6, up7, up8)$;
$up1 = (up9, up10, up11, up12, up13, up14)$;
$up2 = (up15, up16, up17, up18, up19, up20)$;
$up3 = (up21, up22, up23, up24, up25, up26, \mathrm{up27})$;
$up4 = (up28, up29, up30, up31, up32, up33)$;
$up5 = (up34, up35, up36, up37, up38, up39)$;
$up6 = (up40\ up41, up42, up43, up44, up45, up46)$;
$up7 = (up47, up48, up49, up50)$;
$up8 = (up51, up52, up53)$;

Por fim, foram definidas apenas relações de ordem obrigatórios entre os 8 períodos do curso, como pode ser visto na Tabela 9. Desse modo, um período só pode ser completado quanto todas as matérias do período anterior tiverem sido terminadas.

Tabela 9: Relações de ordem dos tópicos de Ciência da Computação

$up1 = obg(up2)$;
$up2 = obg(up3)$;
$up3 = obg(up4)$;
$up4 = obg(up5)$;
$up5 = obg(up6)$;

$up6 = obg(up7)$;

$up7 = obg(up8)$;

A fim de prover uma forma de visualização mais intuitiva, a Figura 21 apresenta a representação gráfica da taxonomia dos tópicos construídos. Por critérios de simplificação, as relações de ordem dessa estrutura não representados na imagem, todavia existe, como dito anteriormente, um fluxo de execução que se inicia na unidade $up1$ (i.e., unidade do tópico Período 1) e termina na unidade $up8$ (i.e., unidade do tópico Período 8). Nesse sentido é importante frisar duas situações impostas por esse tipo de estruturação, tal como segue:

- **Situação 1:** a definição de tópicos para modelar os 8 períodos do curso representa uma alternativa interessante do ponto de vista da organização da estrutura, pois torna seu fluxo de execução muito mais fácil de compreender. Entretanto, a rigidez dessa estruturação apresenta um efeito colateral no comportamento da estrutura, haja vista que nenhum tópico de um período posterior poderá ser percorrido até que todos os subtópicos do tópico anterior sejam percorridos. Em outras palavras, é necessário terminar todas matérias do 1º período para poder cursar as matérias do 2º período e respectivamente até o 8º período;

- **Situação 2:** nenhum dos tópicos referentes as matérias do curso (i.e., as unidades que vão desde $up9$ até $up53$) apresentam qualquer tipo de relação de ordem. Desse modo, as subunidades de um determinado período como, por exemplo, as da unidade $up1$ (i.e., do 1º período) podem ser estudas em qualquer ordem. Entretanto, como frisado na situação anterior, um novo período estará disponível apenas quando todas as subunidades do período anterior tiverem sido completadas.

Tabela 10: Declaração dos tópicos de Ciência da Computação

$up0$ = "Ciência da Computação";	$up27$ = "Empreendedorismo em Informática";
$up1$ = "Periodo 1";	$up28$ = "Teoria e Paradigmas de Linguagem de Programação";
$up2$ = "Periodo 2";	$up29$ = "Engenharia de Software 1";
$up3$ = "Periodo 3";	$up30$ = "Teoria da Computação";
$up4$ = "Periodo 4";	$up31$ = "Cálculo 4";
$up5$ = "Periodo 5";	$up32$ = "Organização e Arquietura de Computadores";
$up6$ = "Periodo 6";	$up33$ = "Probabilidade e Estatística";
$up7$ = "Periodo 7";	$up34$ = "Compiladores";
$up8$ = "Periodo 8";	$up35$ = "Teoria dos Grafos";
$up9$ = "Inglês Instrumenal ";	$up36$ = "Inteligência Artificial 1";
$up10$ = "Programação 1";	$up37$ = "Redes de Computadores 1";
$up11$ = "Laboratório de Programação";	$up38$ = "Banco de Dados 1";
$up12$ = "Cálculo 1";	$up39$ = "Sistemas Operacionais";
$up13$ = "Geometria Analítica";	$up40$ = "Projeto e Análise de Algoritmos";
$up14$ = "Introdução à Computação";	$up41$ = "Engenharia de Software 2";
$up15$ = "Física 1";	$up42$ = "Inteligência Artificial 2";
$up16$ = "Programação 2";	$up43$ = "Redes de Computadores 2";
$up17$ = "Linguagens Formais e Autômatos";	$up44$ = "Banco de Dados 2";
$up18$ = "Cálculo 2";	$up45$ = "Métodos Numéricos";
$up19$ = "Algebra Linear";	$up46$ = "Computação Gráfica";
$up20$ = "Matemática Discreta";	$up47$ = "Interação Homem − Máquina";
$up21$ = "Física 3";	$up48$ = "Sistemas Distribuídos";
$up22$ = "Programação 3";	$up49$ = "Introdução ao Direito";
$up23$ = "Metodologia da Pesquisa e do Trabalho Científico";	$up50$ = "Introdução à Administração";
$up24$ = "Cálculo 3";	$up51$ = "TCC";
$up25$ = "Circuitos Digitais";	$up52$ = "Computador Sociedade e Ética";
$up26$ = "Lógica Aplicada à Computação";	$up53$ = "Gerência de Projetos";

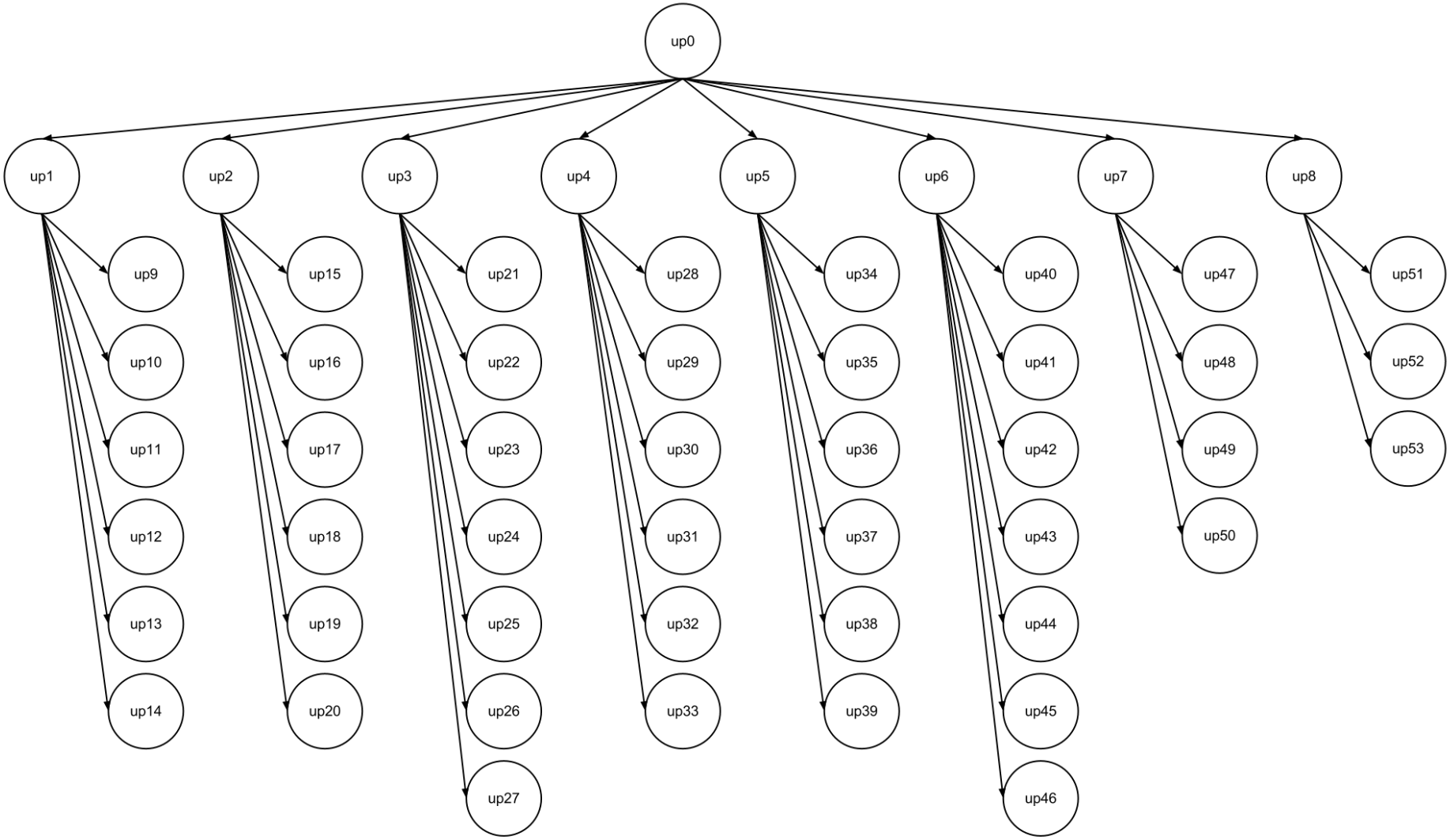


Figura 21: Taxonomia dos tópicos de Ciência da Computação

5.1.2. Estruturação do *Curriculum:* uma abordagem centrada nas relações de ordem

Tal como na seção anterior, visa-se construir uma visão curricular para o curso de Ciência da Computação utilizando o modelo de estruturação de *curriculum*. Contudo, diferentemente da estruturação anterior, essa estruturação enfatiza as relações de ordem do curso. Assim, utilizando-se a linguagem para a estruturação do *curriculum* pode-se chegar à estrutura definida na Tabela 12. Essa estrutura é formada por uma unidade raiz, que representa o curso de Ciência da Computação em si, e por mais 45 unidades pedagógicas, que representam as matérias obrigatórias do curso.

Nesse tipo de estruturação não estão sendo levados em consideração os períodos do curso e, desse modo, a estruturação ganha uma maior flexibilidade. Portanto, existem apenas dois níveis de profundidade na árvore de tópicos (vide Tabela 11): o primeiro diz respeito à unidade raiz, que é o conceito mais geral do *curriculum;* o segundo que é composto por todas as matérias do curso. Desse modo, as relações de ordem podem ser mais facilmente definidas e, além disso, a estrutura se mostra mais adequada (vide Tabela 13). Nesse tipo, de estruturação o estudante teria mais liberdade para empregar seu ritmo de estudo, não estando este sendo regido por uma questão temporal, mas sim por uma questão de relações de pré-requisito definidas pelo professor.

Tabela 11: Taxonomia dos tópicos de Ciência da Computação

$$up0 = (up1, up2, up3, up4, up5, up6, up7, up8, up9, up10, up11, up12, up13, up14,$$
$$up15, up16, up17, up18, up19, up20, up21, up22, up23, up24, up25, up26,$$
$$up27, up28, up29, up30, up31, up32, up33, up34, up35, up36, up37, up38,$$
$$up39, up40, up41, up42, up43, up44, up45);$$

Tabela 12: Declaração dos tópicos de Ciência da Computação

$up0 =$ "Ciência da Computação";	$up23 =$ "Cálculo 4";
$up1 =$ "Inglês Instrumenal";	$up24 =$ "Organização e Arquietura de Computadores";
$up2 =$ "Programação 1";	$up25 =$ "Probabilidade e Estatística";
$up3 =$ "Laboratório de Programação";	$up26 =$ "Compiladores";
$up4 =$ "Cálculo 1";	$up27 =$ "Teoria dos Grafos";
$up5 =$ "Geometria Analítica";	$up28 =$ "Inteligência Artificial 1";
$up6 =$ "Introdução à Computação";	$up29 =$ "Redes de Computadores 1";
$up7 =$ "Física 1";	$up30 =$ "Banco de Dados 1";
$up8 =$ "Programação 2";	$up31 =$ "Sistemas Operacionais";
$up9 =$ "Linguagens Formais e Autômatos";	$up32 =$ "Projeto e Análise de Algoritmos";
$up10 =$ "Cálculo 2";	$up33 =$ "Engenharia de Software 2";
$up11 =$ "Algebra Linear";	$up34 =$ "Inteligência Artificial 2";
$up12 =$ "Matemática Discreta";	$up35 =$ "Redes de Computadores 2";
$up13 =$ "Física 3";	$up36 =$ "Banco de Dados 2";
$up14 =$ "Programação 3";	$up37 =$ "Métodos Numéricos";
$up15 =$ "Metodologia da Pesquisa e do Trabalho Científico";	$up38 =$ "Computação Gráfica";
$up16 =$ "Cálculo 3";	$up39 =$ "Interação Homem − Máquina";
$up17 =$ "Circuitos Digitais";	$up40 =$ "Sistemas Distribuídos";
$up18 =$ "Lógica Aplicada à Computação";	$up41 =$ "Introdução ao Direito";
$up19 =$ "Empreendedorismo em Informática";	$up42 =$ "Introdução à Administração";
$up20 =$ "Teoria e Paradigmas de Linguagem de Programação";	$up43 =$ "TCC";
$up21 =$ "Engenharia de Software 1";	$up44 =$ "Computador Sociedade e Ética";
$up22 =$ "Teoria da Computação";	$up45 =$ "Gerência de Projetos";

Tabela 13: Relações de ordem dos tópicos de Ciência da Computação

$up2 = obg(up8, up9);$	
$up4 = obg(up10);$	$p20 = obg(up26);$
$up5 = obg(up11);$	$up21 = obg(up33);$
$up6 = obg(up17);$	$up22 = obg(up27);$
$up7 = obg(up13);$	$up23 = obg(up38);$
$up8 = obg(up14, up32);$	$up24 = obg(up29, up31);$
$up9 = obg(up22);$	$up25 = obg(up28);$
$up10 = obg(up16);$	$up26 = obg(up33);$
$up11 = obg(up38);$	$up28 = obg(up34);$
$up12 = obg(up18);$	$up29 = obg(up35);$
$up13 = obg(up29);$	$up30 = obg(up33, up36);$
$up14 = obg(up21, up28);$	$up33 = obg(up39);$
$up16 = obg(up23);$	$up34 = obg(up39);$
$up17 = obg(up24);$	$up35 = obg(up40);$
$up18 = obg(up25, up30);$	$up41 = obg(up44);$
$up19 = obg(up41, up42);$	$up42 = obg(up45);$

A fim de prover uma forma de visualização mais intuitiva, a Figura 22 apresenta a representação gráfica dos relacionamento de ordem dos tópicos construídos. Por critérios de simplificação, as relações de taxonomia dessa estrutura não representados na imagem, todavia existe, como dito anteriormente, apenas uma unidade raiz $up0$ e suas unidades filhas que são todas as outras unidades. Nesse sentido é importante frisar duas situações impostas por esse tipo de estruturação, tal como segue:

- **Situação 1:** como pode ser visto o curso é composto por alguns troncos principais como, por exemplo, o ramo que se inicia com a unidade $up2$ (i.e., unidade de Programação 1) e que culmina nas unidades $up27, up32$ e $up39$ (i.e., as unidades de Teoria dos Grafos, Projeto e Análise de Algoritmos e Interação Homem – Máquina, respectivamente);

- **Situação 2:** algumas unidades aparecem mais isoladas como, por exemplo, as unidades $up1, up3$ e $up15$ (i.e., as unidades de Inglês Instrumental, Laboratório de Programação e Metodologia da Pesquisa e do Trabalho Científico, respectivamente).

Tal isolamento poderia levantar a alguns questionamentos sobre a organização ou pelo menos a integração dessas disciplinas para com o curso como um todo;

- **Situação 3:** algumas unidades como, por exemplo, as unidades $up37$ e $up43$ (i.e., as unidades de Métodos Numéricos e TCC, respectivamente) se mostram totalmente isoladas e sem qualquer tipo de pré-requisito definido. Nesse sentido, acredita-se que tal estruturação não reflete a realidade dessas unidades, haja vista que é necessária uma carga de conhecimento prévio, de unidades que compõem o próprio curso, antes desses temas serem abordados.

Assim, é possível notar que com esse tipo de estruturação foi possível ter um retrato mais fiel sobre a organização do curso como um todo. Tal modelagem poderia servir, por exemplo, para uma possível discussão acerca da grade curricular do curso em si.

5.2. Caso 2: Aprendizagem de Máquina

O presente caso investiu na estruturação do curso de Aprendizagem de Máquina ministrado pelo professor Andrew Ng da Universidade *Standford*[13]. Para essa estruturação investiu-se em uma abordagem centrada na taxonomia, tal como definido pelo próprio professor Andrew. Além disso, investiu-se na construção da rede de Petri visando a simulação da estrutura de tópicos.

5.2.1. Estruturação do *Curriculum*

Com base na estrutura proposta pelo professor Andrew Ng, pode-se fixar o domínio de Aprendizagem de Máquina e construir uma visão curricular para o mesmo utilizando o modelo de estruturação. Assim, utilizando-se a linguagem para a estruturação do *curriculum* pode-se chegar à estrutura definida na Tabela 14. Essa estrutura é formada por uma unidade

[13] Disponível em https://www.coursera.org/course/ml.

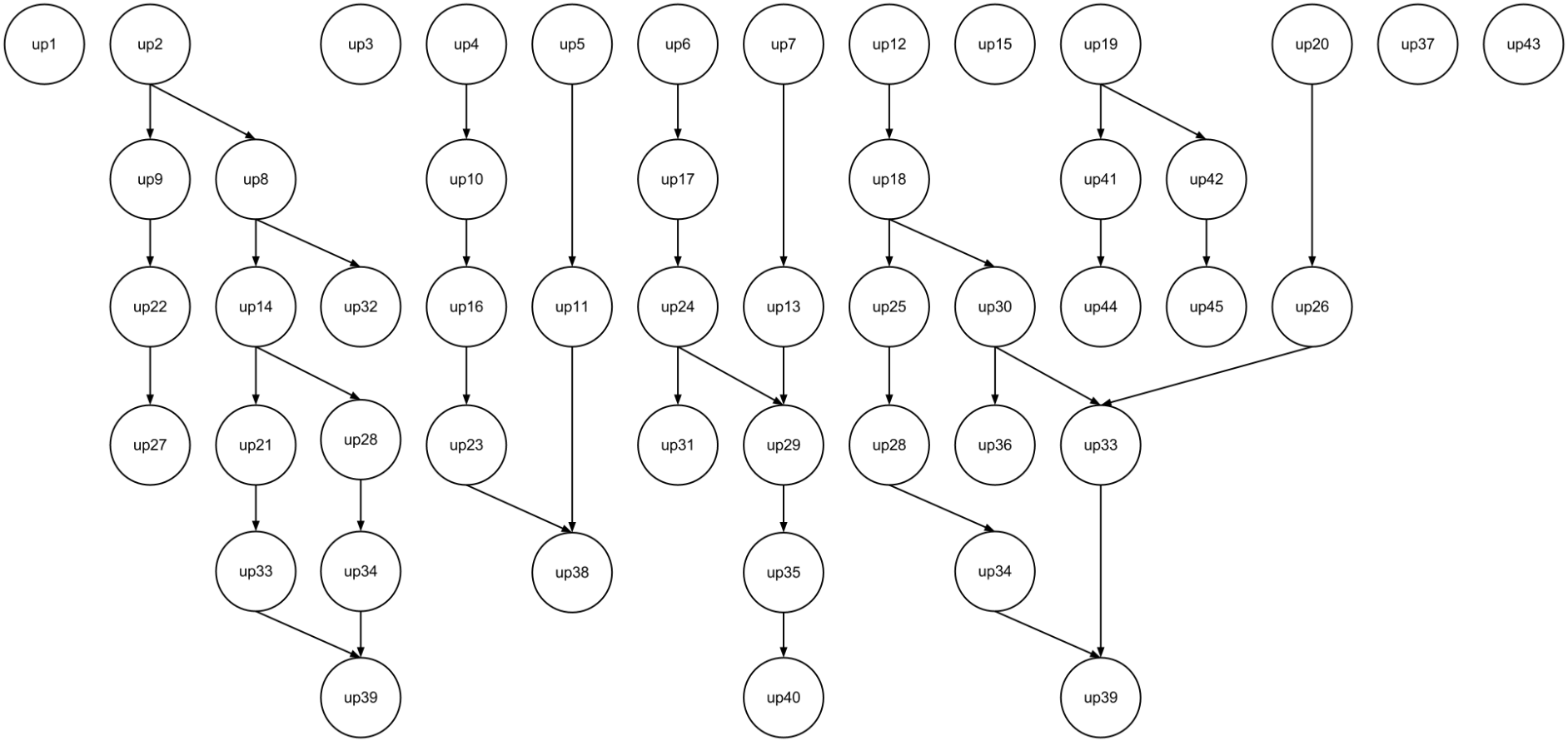


Figura 22: Relações de ordem dos tópicos de Ciência da Computação

raiz, que representa o curso de Aprendizagem de Máquina em si, e por mais 19 unidades pedagógicas, que representam os conceitos a serem abordados no curso.

5.2.2. Estruturação do *Curriculum*

Com base na estrutura proposta pelo professor Andrew Ng, pode-se fixar o domínio de Aprendizagem de Máquina e construir uma visão curricular para o mesmo utilizando o modelo de estruturação. Assim, utilizando-se a linguagem para a estruturação do *curriculum* pode-se chegar à estrutura definida na Tabela 14. Essa estrutura é formada por uma unidade raiz, que representa o curso de Aprendizagem de Máquina em si, e por mais 19 unidades pedagógicas, que representam os conceitos a serem abordados no curso.

Tabela 14: Declaração dos tópicos de Aprendizagem de Máquina

$up0 =$ "Aprendizagem de Máquina";	$up10 =$ "Revisão álgebra linear";
$up1 =$ "Semana1";	$up11 =$ "Regressão linear múltiplas variáveis";
$up2 =$ "Semana 2";	$up12 =$ "Tutorial Octave";
$up3 =$ "Semana 3";	$up13 =$ "Regressão logística";
$up4 =$ "Semana 4";	$up14 =$ "Regularização";
$up5 =$ "Semana 5";	$up15 =$ "Redes neurais: Representação";
$up6 =$ "Semana 6";	$up16 =$ "Redes neurais: Aprendizagem";
$up7 =$ "Semana 7";	$up17 =$ "Conselhos para aplicar aprend.. de máquina";
$up8 =$ "Introdução";	$up18 =$ "Projeto de sistemas de aprend. de máquina";
$up9 =$ "Regressão linear uma variável";	$up19 =$ "Aprendizagem SVM";

O curso foi estruturado em semanas e, por conseguinte, foram definidas 7 unidades que visam apenas estruturar o curso segundo esse período de tempo. Assim, como apresentado na Tabela 15, as unidades que vão desde $up1$ até $up7$ tem unicamente a função de isolar as matérias do curso segundo o período previsto.

Tabela 15: Taxonomia dos tópicos de Aprendizagem de Máquina

$up0 = (up1, up2, up3, up4, up5, up6, up7);$
$up1 = (up8, up9, up10);$
$up2 = (up11, up12);$
$up3 = (up13, up14);$
$up4 = (up15);$
$up5 = (up16);$
$up6 = (up17, up18);$
$up7 = (up19);$

Assim como na primeira abordagem do curso de Ciência da Computação foram definidas apenas relações de ordem obrigatórias entre as 7 semanas do curso, como pode ser visto na Tabela 16. Todavia é importante destacar uma situação particular na estruturação desse curso:

- **Situação 1:** para esse curso o professor Andrew definiu dois tópicos como opcionais e, nesse sentido, é possível utilizar o relacionamento opcional definido no modelo de estruturação. Assim, as unidades $up10$ e $up12$ (i.e., Revisão Álgebra Linear e Tutorial *Octave*, respectivamente) não são obrigatórias para completar o curso de Aprendizagem de Máquina. Assim, na primeira semana de curso, quando as unidades $up8$ e $up9$ (i.e., Introdução e Regressão Linear com uma variável, respectivamente) tiverem sido terminadas os tópicos da segunda semana estarão disponíveis mesmo sem o término da unidade $up10$.

A fim de prover uma forma de visualização mais intuitiva, a Figura 23 e a Figura 24 apresentam, respectivamente, a representação gráfica da taxonomia e dos relacionamentos de ordem construídos.

Tabela 16: Relações de ordem dos tópicos de Aprendizagem de Máquina

$up1 = obg(up2);$	
$up2 = obg(up3);$	$up8 = obg(up9);$
$up3 = obg(up4);$	$up9 = opc(up10);$
$up4 = obg(up5);$	$up11 = opc(up12);$
$up5 = obg(up6);$	$up13 = obg(up14);$
$up6 = obg(up7);$	$up17 = obg(up18);$

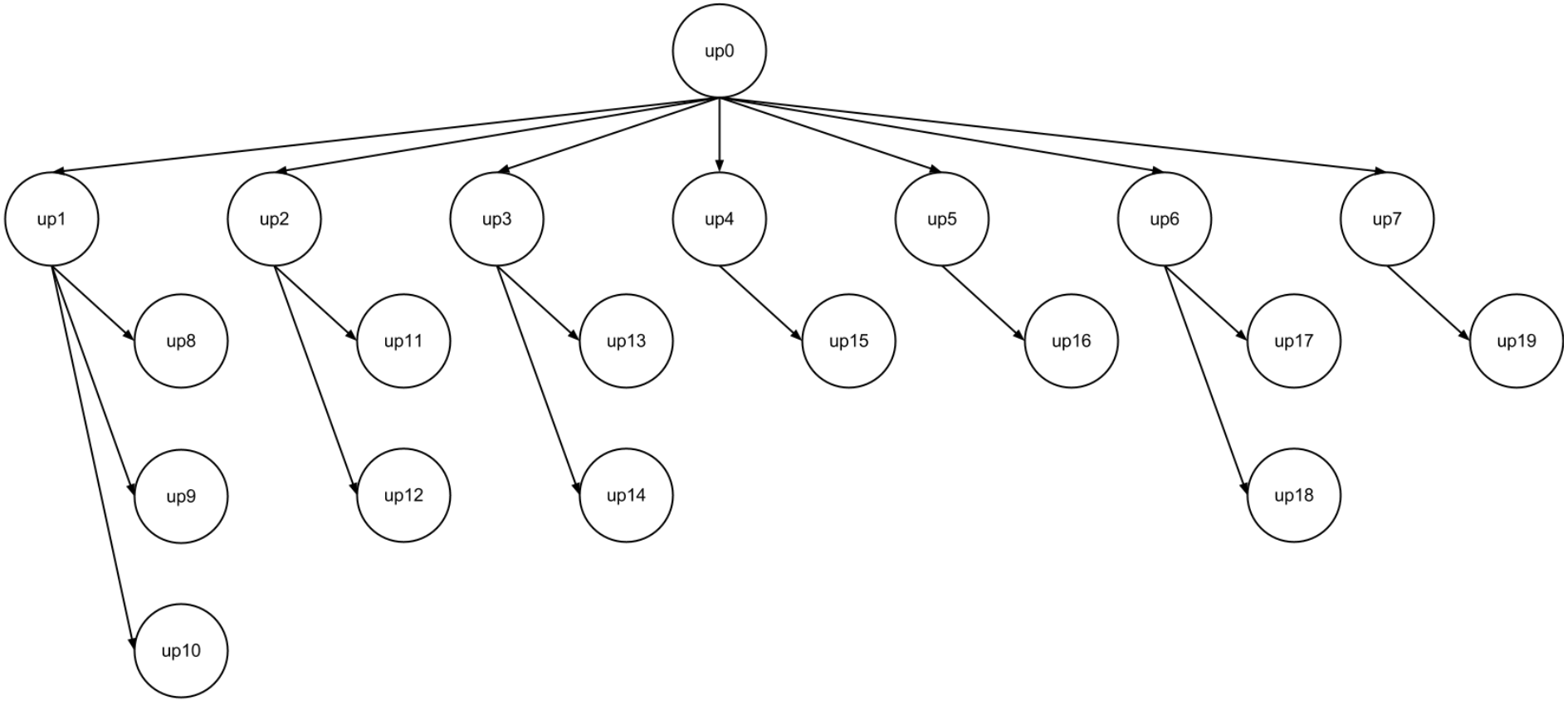


Figura 23: Taxonomia dos tópicos de Ciência da Computação

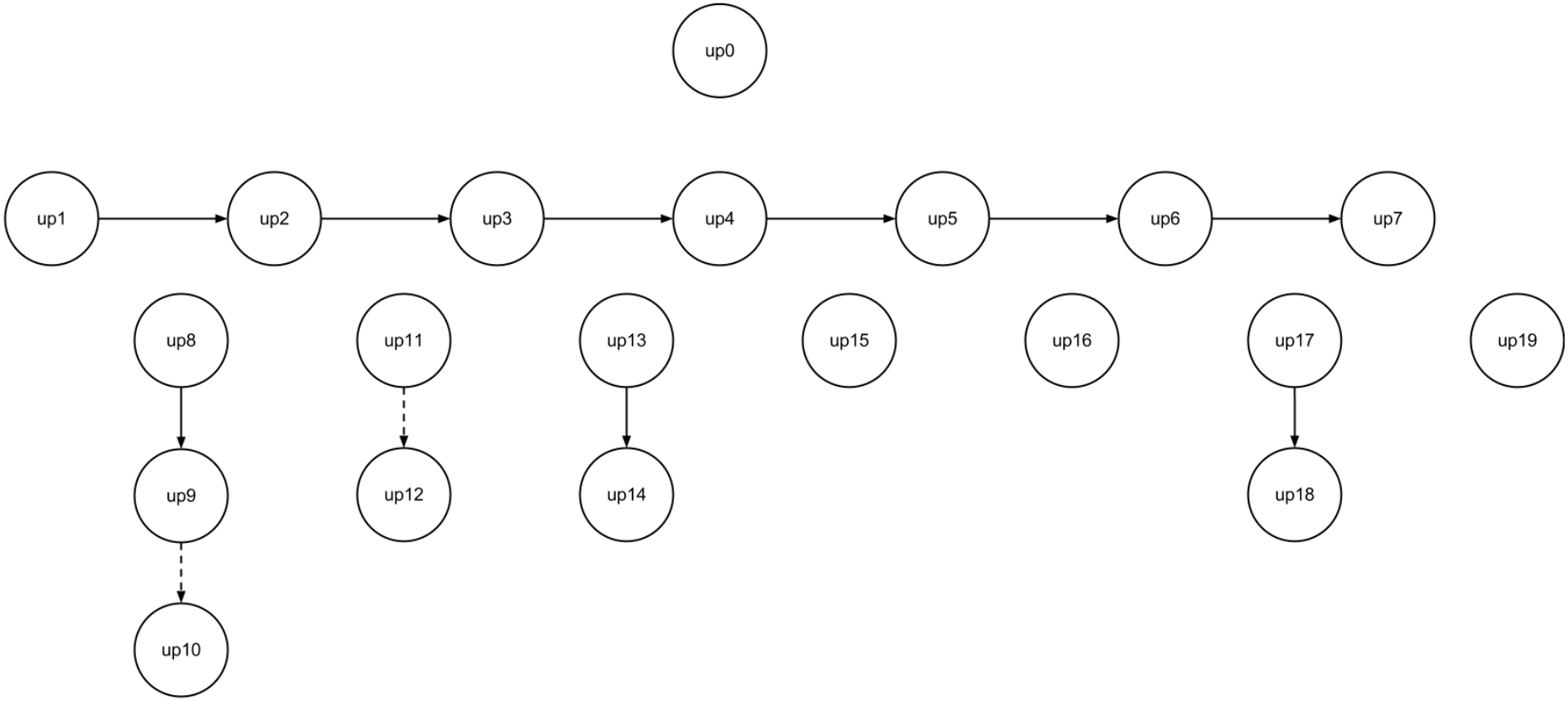


Figura 24: Relações de ordem dos tópicos de Ciência da Computação

5.2.3. Verificação do *Curriculum*

Com base na estrutura de grafo construída anteriormente é possível, a partir dos algoritmos definidos no modelo de transformação, derivar uma rede de Petri correspondente. Nesse sentido, tal como pode ser visto na Figura 25, a rede de Petri resultante da estrutura de *curriculum* apresenta o fluxo de execução do curso, ou seja, apresenta o caminhamento na árvore de tópicos levando em consideração tanto as relações taxonômicas quanto as relações de ordem.

Para a verificação da estrutura foi simulada uma situação com cinco alunos distintos. Basicamente, cada estudante vai navegando na estrutura de tópicos de forma sequencial, desde a unidade inicial $up8$ até a unidade final $up0$. Entretanto, é importante destacar algumas situações, tal como segue:

- **Situação 1**: o Aluno 2 encontra-se em uma separação ou bifurcação, sendo inserido em uma situação de tomada de decisão. Como o caminhamento da unidade $up11$ (i.e., regressão linear com múltiplas variáveis) para a unidade $up12$ (i.e., tutorial *Octave*) foi definido como opcional, o aluno percorrer dois caminhos: (1) percorrer a unidade opcional e depois se dirigir para a unidade $up2$; ou (2) se dirigir diretamente para a unidade $up2$ sem passar por $up12$. Para efeitos de verificação, qualquer um dos dois caminhos é válido;

- **Situação 2:** o Aluno 3 passou por uma situação análoga ao da situação anterior e, nesse caso, resolveu percorrer a unidade opcional. Nesse sentido, é possível pela simulação apresentada que o Aluno 3 encontra-se exatamente na unidade $up10$ (i.e., revisão álgebra linear).

Para as demais situações, que são representadas pelo Aluno 1, pelo Aluno 4 e pelo Aluno 5, enfatiza-se apenas que, nesse caso, os alunos devem obrigatoriamente seguir o fluxo de execução que é único. Além disso, o ciclo de cada aluno é isolado dos demais e, assim, não existe qualquer tipo de influência entre um estudante e outro.

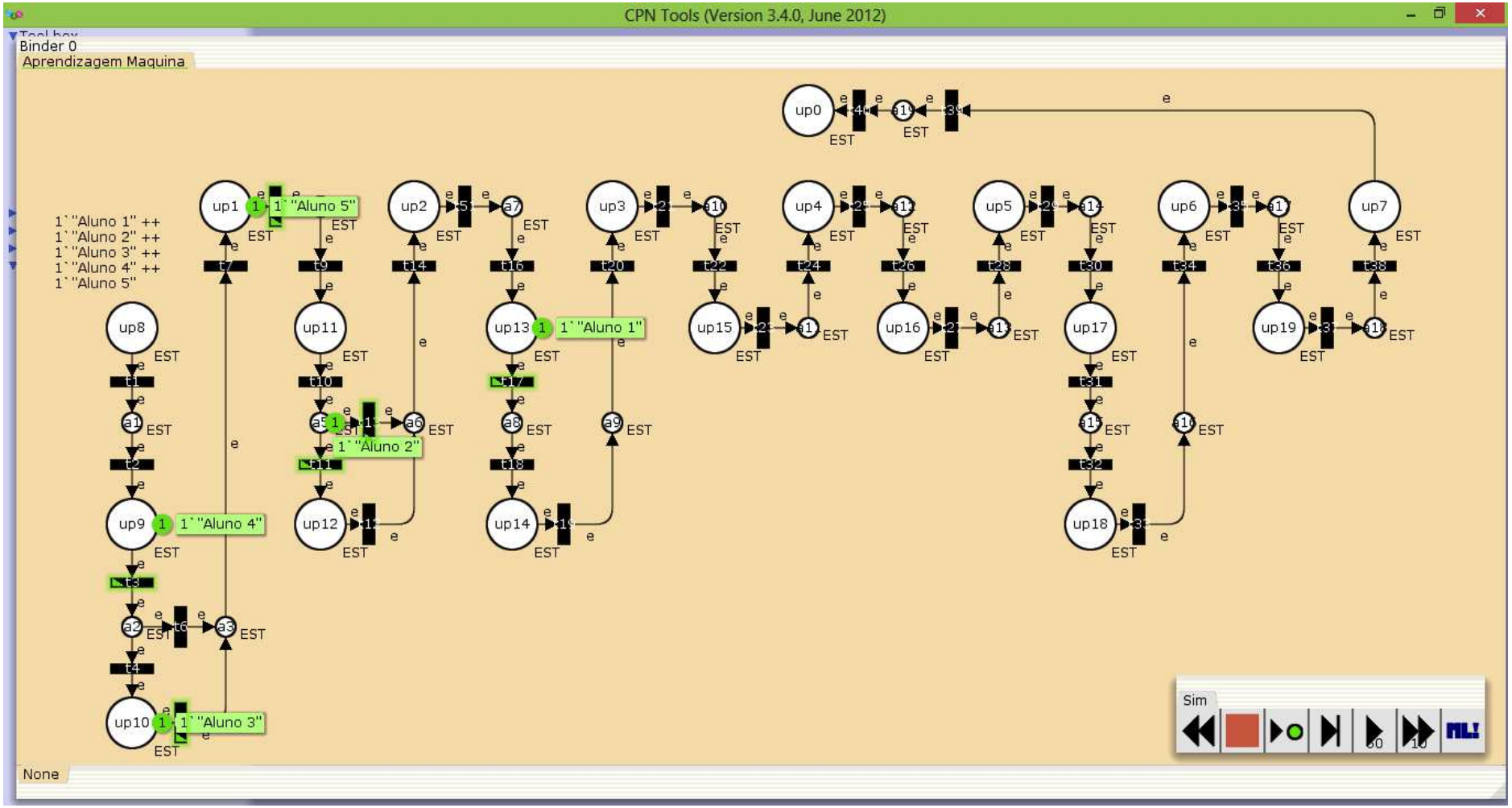


Figura 25: Rede de Petri do *curriculum* de Aprendizagem de Máquina

5.3. Caso 3: Lógica Computacional

O presente caso investiu na estruturação do curso de Lógica Computacional ministrado pelo professor Evandro de Barros Costa da Universidade Federal de Alagoas. Para essa estruturação investiu-se em uma abordagem centrada na taxonomia, tal como definido pelo próprio professor Andrew. Além disso, investiu-se na construção da rede de Petri visando a simulação da estrutura de tópicos. Por fim, investiu-se no desenvolvimento dos agentes de tutores.

5.3.1. Estruturação do *Curriculum*

Com base na estrutura proposta pelo professor Evandro de Barros Costa, pode-se fixar o domínio de Lógica Computacional e construir uma visão curricular para o mesmo utilizando o modelo de estruturação. A Figura 26 e a Figura 27 apresentam, respectivamente, os tópicos e a taxonomia da estrutura de unidades pedagógicas. Essa estrutura é formada por uma unidade raiz, que representa o curso de Lógica Computacional em si, e por mais 20 unidades pedagógicas, que representam os conceitos a serem abordados no curso.

Para as relações de ordem foram definidas apenas relações obrigatórias entre os tópicos do curso, como pode ser visto na Figura 28. Entretanto, é importante ressaltar os pares formadas pelas unidades $up9$ e $up10$, $up17$ e $up18$ e $up19$ e $up20$, que não possuem qualquer relação de ordem. Nesse tipo de situação, o modelo define que as unidades podem ser percorridas em qualquer ordem, todavia frisa-se que todas essas unidades devem ser percorridas para que a unidade pai (e.g., a unidade $up3$) possa ser terminada.

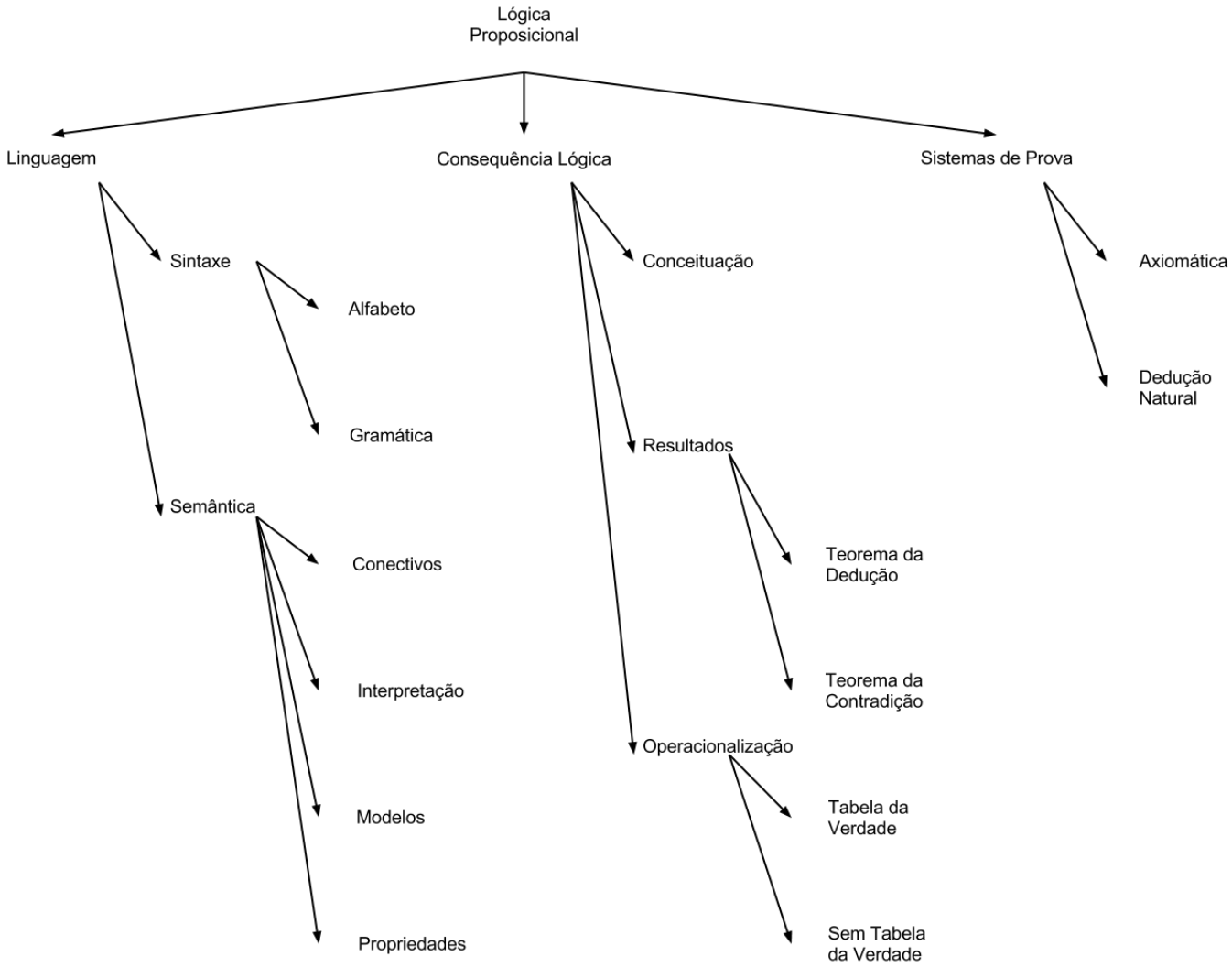


Figura 26: Tópicos de Lógica Computacional

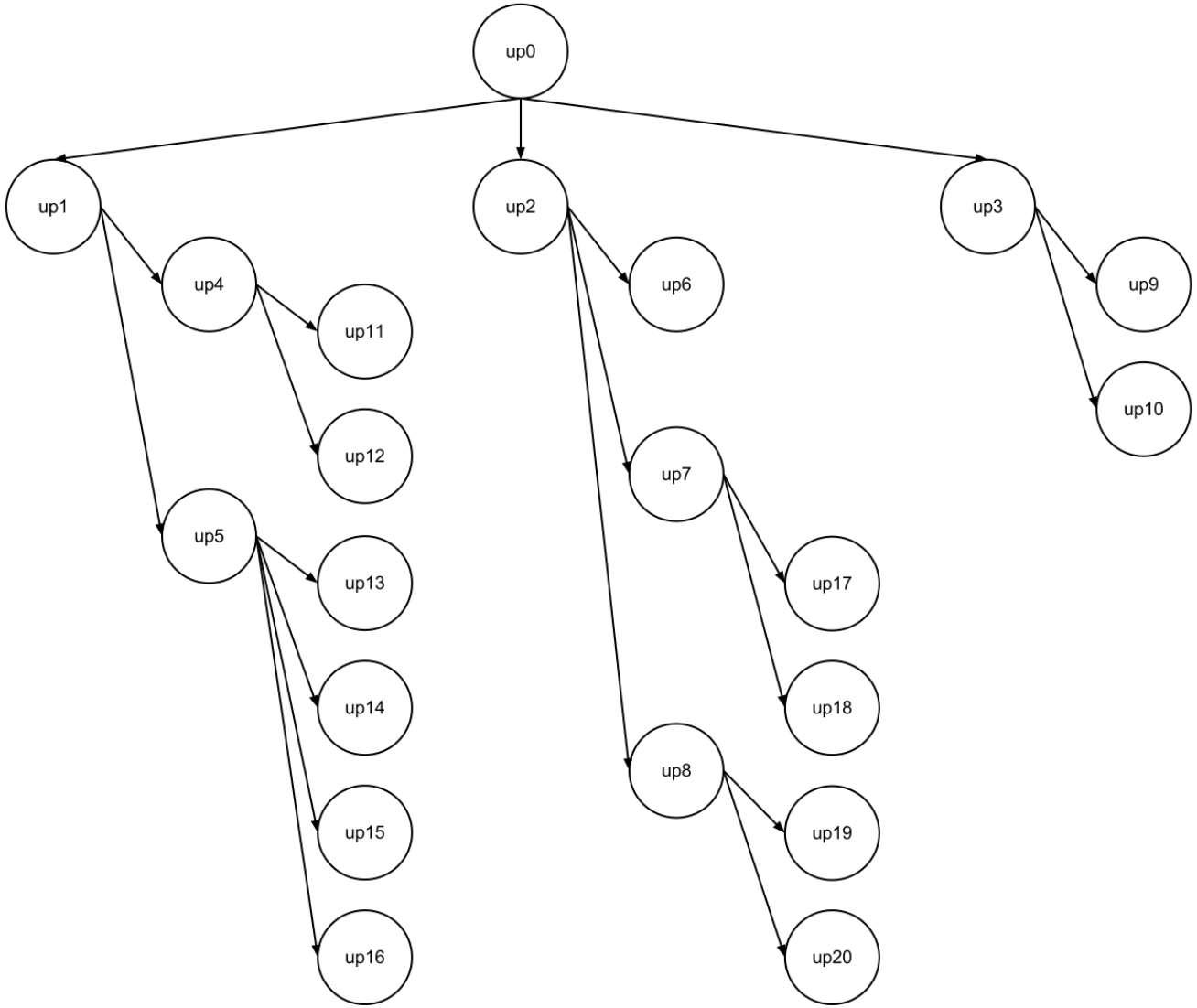


Figura 27: Taxonomia dos tópicos de Lógica Computacional

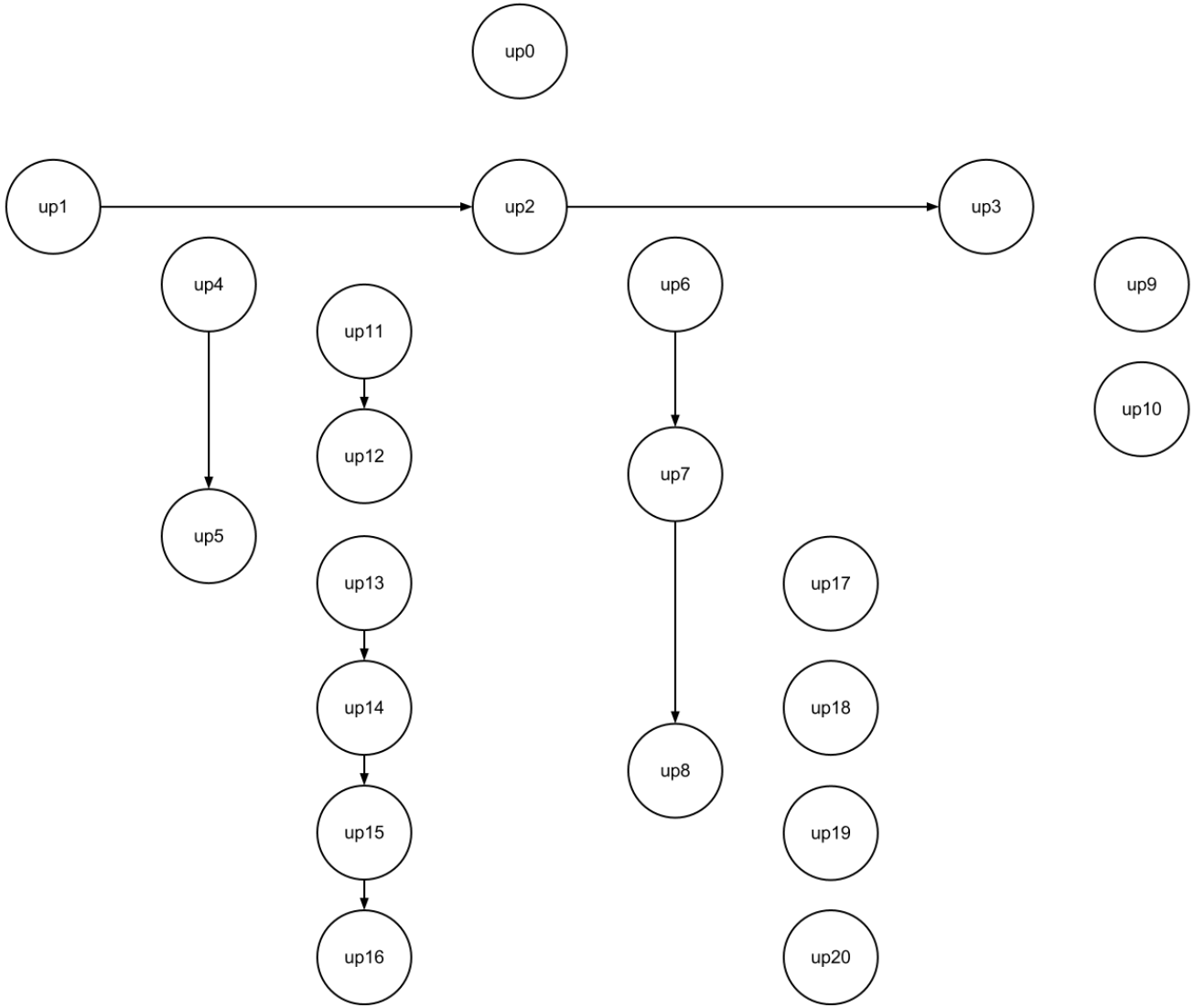


Figura 28: Relações de ordem dos tópicos de Lógica Computacional

5.3.2. Verificação do *Curriculum*

Com base na estrutura de grafo construída anteriormente foi construída uma rede de Petri correspondente. Nesse sentido, tal como pode ser visto na Figura 25, a rede de Petri resultante da estrutura de *curriculum* apresenta o fluxo de execução do curso, ou seja, apresenta o caminhamento na árvore de tópicos levando em consideração tanto as relações taxonômicas quanto as relações de ordem definidas pelo professor. Assim, visa-se simular a execução dessa estrutura no intuito de verificar se o comportamento apresentado pela mesma condiz com a proposta imaginada pelo autor.

Para a verificação da estrutura foi simulada uma situação com cinco alunos distintos. Basicamente, cada estudante vai navegando na estrutura de tópicos de forma sequencial, desde a unidade inicial $up11$ até a unidade final $up0$. Nesse sentido é importante frisar algumas situações, tal como segue:

- **Situação 1**: o Aluno 2 encontra-se em uma separação ou bifurcação entre as unidades $up17$ e $up18$ (i.e., os tópicos de Teorema da Dedução e Teorema da Contradição, respectivamente). Nessa situação não existe nenhum tipo de ordem previamente definida e, desse modo, qualquer uma das duas unidades pode ser percorrida primeiro. Para efeitos de verificação, serão criadas duas instâncias do Aluno 2 e cada uma delas será alocada nas unidades aqui discutidas. Em suma, tal separação pode ser traduzida como se o Aluno 2 estivesse estudando os dois tópicos ao mesmo tempo, podendo hora estar estudando um e hora estar estudando o outro;

- **Situação 2:** o Aluno 3 encontra-se em uma situação de espera para poder se encaminhar para a unidade $up3$ (i.e., o tópico de Sistemas de Provas). Nessa situação o aluno encontra-se impedido de continuar seus estudos, pois existem pré-requisitos que ainda não terminados. Mais especificamente, para a unidade $up3$ existem dois pré-requisitos: (1) a unidade $up9$, que representa o tópico Axiomática; e (2) a unidade $up10$, que representa o tópico Dedução Natural. Assim, mesmo tendo terminado a unidade $up9$ é possível notar que o aluno ainda não iniciou a unidade $up10$ e, como tal, será necessário termina-la para poder continuar o caminhamento na estrutura;

- **Situação 3:** o Aluno 4 encontra-se em uma situação de espera para poder se encaminhar para a unidade $up8$ (i.e., o tópico de Operacionalização). Essa situação é análoga ao apresentado na situação 2, todavia, diferentemente da situação anterior, o aluno terminou de percorrer uma das unidades (i.e., a unidade $up19$, que representa o tópico de Tabela da Verdade) e já iniciou a outra unidade (i.e., a unidade $up20$, que representa o tópico Sem Tabela da Verdade). Nesse caso, assim que o aluno terminar de percorrer a unidade $up20$ a unidade $up8$ estará disponível. Para efeitos de verificação, quando as duas instâncias do Aluno 4 serão reduzidas a apenas uma, que continuará a exploração da estrutura de tópicos a partir da unidade $up8$.

Para as demais situações, que são representadas pelo Aluno 1 e pelo Aluno 5, enfatiza-se apenas que, nesse caso, os alunos devem obrigatoriamente seguir o fluxo de execução que é único. Além disso, frisa-se novamente que o ciclo de cada aluno é isolado dos demais e, assim, não existe qualquer tipo de influência entre um estudante e outro.

5.3.3. Identificação dos Agentes

Levando em consideração a taxonomia da estrutura definida, deverão ser construídos 20 Agentes Tutores, um para cada tópico do curso. Contudo, caberia ao Engenheiro de *Software* avaliar se tal estruturação dos agentes se mostra interessante do ponto de vista computacional. Nesse sentido, é possível destacar as seguintes situações:

- **Situação 1:** o agente encontra-se sobrecarregada, ou seja, o tópico que está sob a sua responsabilidade é muito geral e engloba uma carga de conhecimento muito grande. Nesse caso, compete ao Engenheiro de Software sugerir ao Especialista que faça algum tipo de estratificação no tópico em questão e, por conseguinte, refletir essa alteração na estrutura de agentes. Outra alternativa seria definir um conjunto de Agentes Tutores para dividir a responsabilidade desse tópico;

- **Situação 2:** o agente encontra-se subutilizado, ou seja, o tópico que está sob a sua responsabilidade é muito específico e engloba uma carga de conhecimento muito

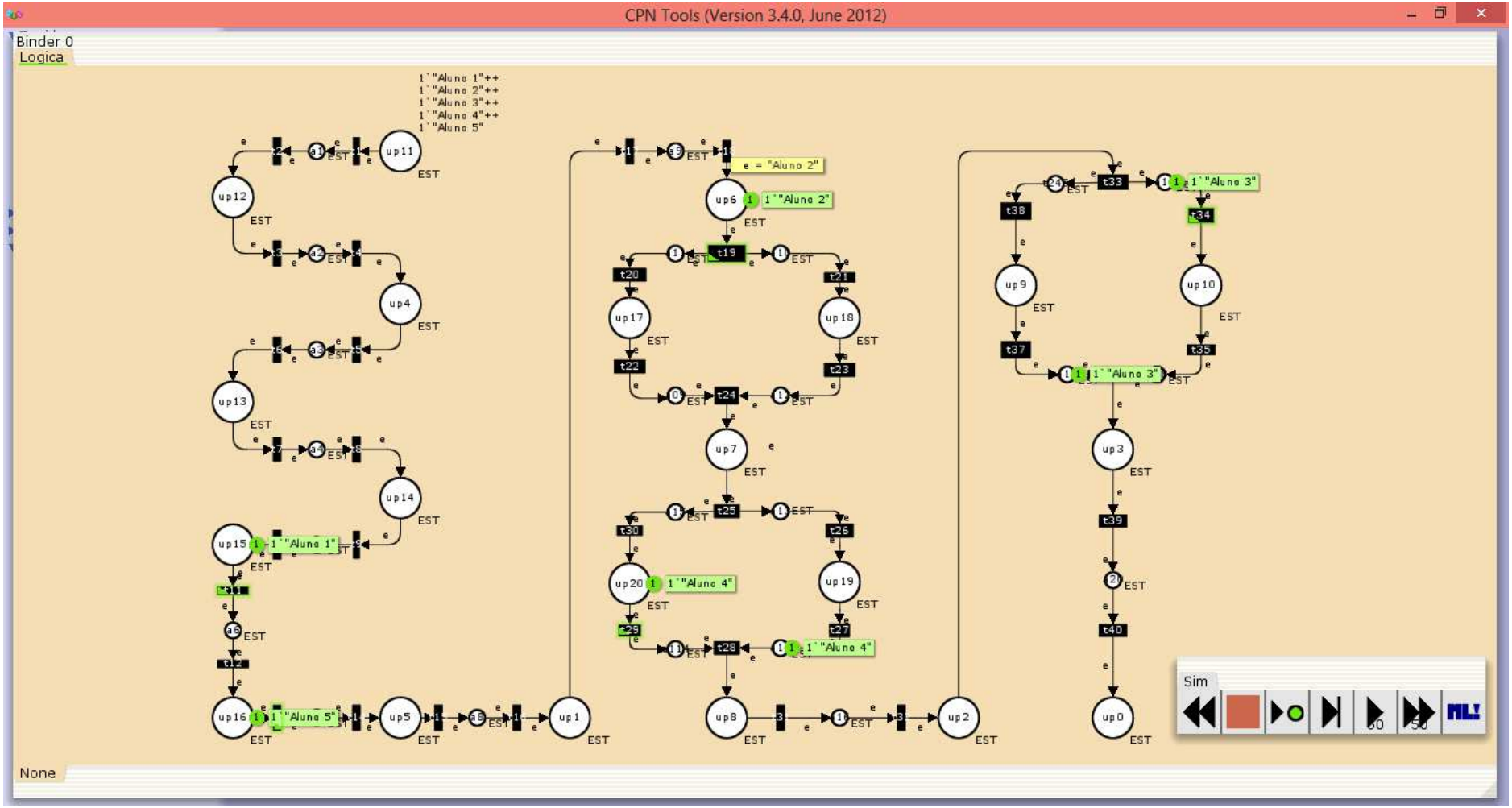


Figura 29: Rede de Petri *curriculum* de Lógica Computacional

baixa. Nesse caso, compete ao Engenheiro de *Software* condensar a estrutura de agentes, definindo agentes que terão responsabilidade em mais de um tópico da estrutura.

Essas duas situações visam ilustrar que a modelagem dos agentes não precisa seguir completamente a estruturação dos tópicos definida pelo Especialista, pois compete ao Engenheiro de *Software* tomar decisões que visam uma maior eficiência do ponto de vista computacional. Assim, o Algoritmo 4 apresentado anteriormente tem a função de definir uma estruturação inicial, que poderá ser alterada conforme novas necessidades forem surgindo.

Sob essa perspectiva, os agentes construídos seguiram completamente a estrutura de tópicos, tal como pode ser visto na Figura 30. No processo de construção os agentes foram separados em contêineres diferentes, assim cada contêiner tem a função de armazenar apenas os Agentes que partilham um mesmo mediador (i.e., agentes irmãos). Esse tipo de estruturação torna o sistema mais organizado e facilita a manutenção do sistema, haja vista que a remoção de não afeta o sistema como um todo, causando pouco impacto nos agentes que não são mediados por ele.

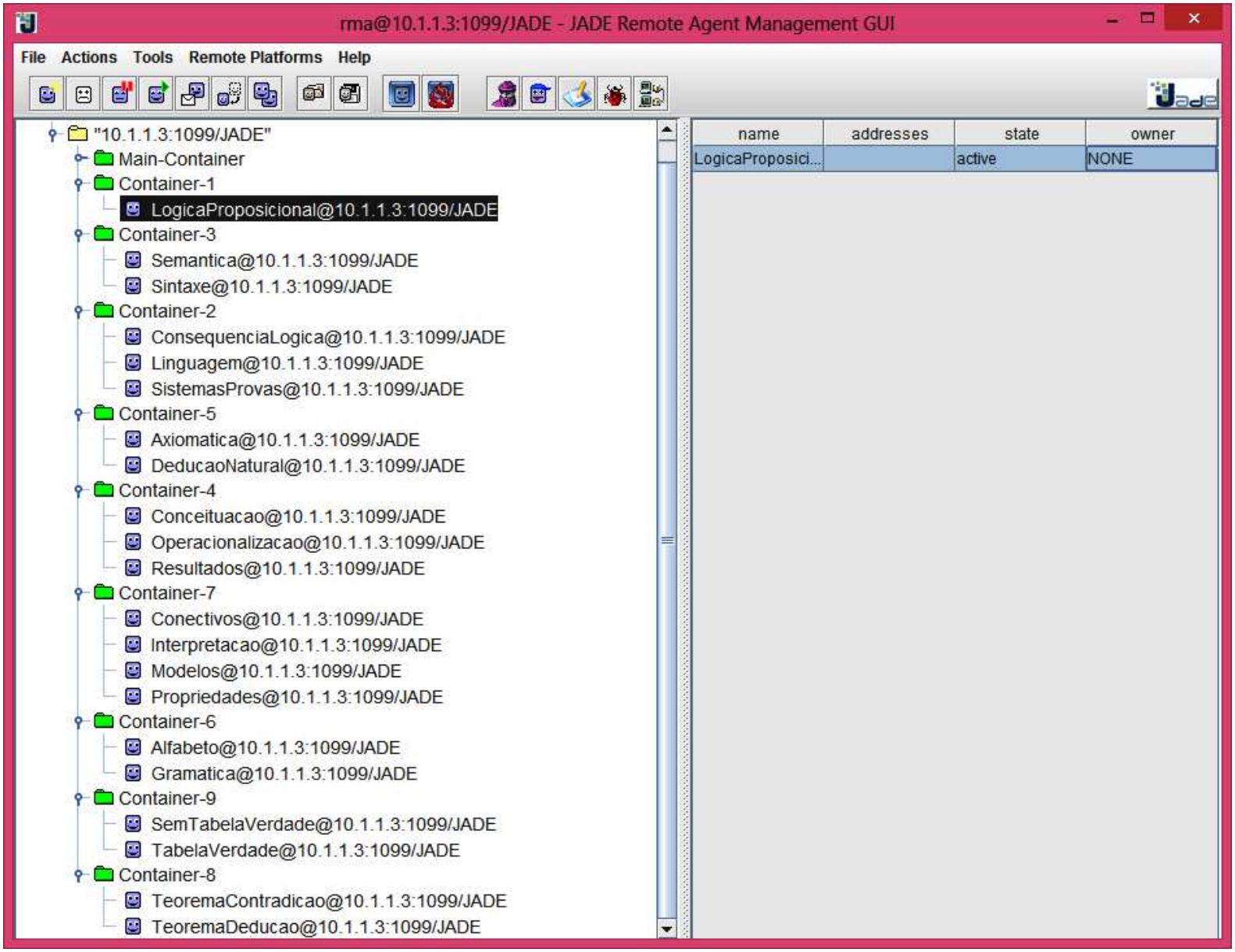


Figura 30: Agentes do *curriculum* de Lógica Computacional

5.4. Considerações Finais

Os estudos de caso apresentados visaram destacar algumas situações que podem ser encontrada cotidianamente. Nesse sentido, buscou-se investir na estruturação de cursos reais, que de certa forma demostraram alguns cenários interessantes. Além disso, acredita-se que tais estudos de caso aliados com os exemplos do curriculum de fração, que foram apresentados no decorrer do documento, apresentaram os modelos construídos de forma satisfatória. Contudo, é importante ressaltar que novos experimentos necessitam ser realizados e, além disso, destacam-se alguns pontos que puderam ser percebidos no decorrer dos experimentos:

- **Modelo de Estruturação do *Curriculum***: esse modelo apresentou uma boa cobertura no que diz respeito aos casos apresentados nessa dissertação. Entretanto, foi possível perceber, sobretudo no caso de Ciência da Computação, que o fator temporal pode ser algo a ser considerado em futuras extensões desse modelo. Outro fator negativo foi a falta de uma ferramenta de autoria para estruturar o *curriculum*. Acredita-se que com uma ferramenta de autoria, novos experimentos executados e novas situações poderão ser descobertas, enriquecendo ainda mais o modelo;

- **Modelo de Verificação**: esse modelo se mostrou interessante, pois com ele foi possível verificar o comportamento das estruturas construídas. Todavia, é importante avaliar que as estruturas apresentadas nos casos não apresentavam uma complexidade muito grande e, como tal, não resultavam em redes de Petri com muitos nós. Nesse sentido, uma abordagem baseada em redes de Petri hierárquicas facilitaria a visualização e tornaria o modelo mais intuitivo. Além disso, uma abordagem temporizada poderia ser utilizada em futuras extensões da proposta;

- **Modelo de Agentes**: a arquitetura de agentes se mostrou adequada e atual, além disso, apresentou uma interligação entre conceitos de Inteligência Artificial e da Engenharia de Software, com a utilização de padrões de projeto. Contudo, novos testes devem ser realizados para avaliar os modelos de forma mais pragmática.

6. CONCLUSÕES E PERSPECTIVAS FUTURAS

Esta dissertação constituiu-se em uma proposta sobre Modelos Computacionais de Conhecimento, tomando como base os resultados previamente obtidos pelo modelo *Mathema*. Nesse contexto, foi proposta uma sistemática, dotada de modelos, para a estruturação do conhecimento de domínio, mais especificamente da estrutura curricular. Além disso, no decorrer do processo foram obtidas algumas publicações (e.g.,Silva et al. [32] e Silva et al [33]). Perante as proposta relacionadas, o trabalho apresentou direcionamentos interessantes, sobretudo no que diz respeito à estruturação do *curriculum*. Por fim, espera-se contribuir para a construção de STMs, destacando-se os seguintes pontos:

- **Modelo de Estruturação de Curriculum:** nesse modelo é possível destacar, primeiramente, a formalização apresentada, que poderá ser utilizada de base para futuros trabalhos. Além disso, houve uma maior aproximação com a visão original proposta por de Barros Costa [1]. Por fim, ressalta-se a definição de uma linguagem textual, de uma linguagem gráfica e de um modelo computacional que modelagem essa estrutura;

- **Modelo de Verificação**: nesse modelo é possível destacar a definição das equivalências entre as relações do modelo estruturação de curriculum e o modelo de verificação. Além disso, frisa-se a construção de um algoritmo de transformação para a construção de rede de Petri de modo automático. Por fim, ressalta-se a utilização de um modelo computacional, representado pela ferramenta CPN Tools, para simulação da dinâmica da estrutura;

- **Modelo de Agentes:** nesse modelo é possível destacar a revisão da arquitetura de agentes do *Mathema*. Além disso, frisa-se a reorganização da estruturação dos agentes de SATA, onde foi definida a figura do Agente Gerenciador.

No entanto, a proposta aqui defendida ainda encontra alguns pontos em aberto. Assim, podem-se destacar os seguintes direcionamentos:

- **Ferramentas:** infelizmente a presenta proposta ainda carece de diversos tipos de ferramentas que poderiam facilitar a utilização da abordagem. Nessa perspectiva, destaca-se a adoção de uma ferramenta de autoria gráfica que possa auxiliar professores no processo de estruturação do *curriculum*. Além disso, ressalta-se a necessidade de desenvolver um módulo de verificação integrar uma solução de autoria com os modelos definidos, por exemplo, no CPN *Tools*;

- **Experimentação:** embora tenham sidos apresentados alguns estudos de caso, o modelo ainda carece de experimentação, sobretudo, com professores. Nesse sentido, espera-se que com o desenvolvimento de uma ferramenta de autoria, tal problema possa ser resolvido;

- **Formalização:** os modelos apresentados nessa proposta dizem respeito a apenas uma parte do modelo de estruturação de conhecimento do *Mathema*. Desse modo, novos investimento podem ser feitos visando a formalização de modelos que contemplem a proposta em sua totalidade.

REFERÊNCIAS BIBLIOGRÁFICAS

1. DE BARROS COSTA, E. **Um modelo de ambiente interativo de aprendizagem baseado numa arquitetura multi-agentes**. Universidade Federal da Paraíba. Campina Grande. 1997. Tese de Doutorado.

2. ANDERSON, J. R.; BOYLE, C. F.; REISER, B. J. Intelligent Tutoring Systems. **Science**, v. 228, n. 4698, p. 456-462, Abril 1985.

3. SELF, J. A. Student models in computer-aided instruction. **International Journal of Man-Machine Studies**, v. 6, n. 2, p. 261-276, Março 1974.

4. KOEDINGER, K. R. et al. Intelligent Tutoring Goes to School in te Big City. **Internation Journal of Artificial Intelligence in Education**, v. 228, n. 1, p. 30-43, 1997.

5. FRASSON, C.; AIMEUR, E. Designing a multi-strategic intelligent tutoring system for training in industry. **Computers in Industry**, v. 37, n. 2, p. 153-167, Setembro 1998.

6. BUTZ, B. P.; DUARTE, M.; MILLER, S. M. An intelligent tutoring system for circuit analysis. **IEEE Transactions on Education**, v. 49, n. 2, p. 216-223, Maio 2006.

7. ALEVEN, V.; MCLAREN, B. M.; SEWALL, J. Scaling Up Programming by Demonstration for Intelligent Tutoring Systems Development: An Open-Access Web Site for Middle School Mathematics Learning. **IEEE Transactions on Learning Technologies**, v. 2, n. 2, p. 64-78, Maio 2009.

8. DE BARROS COSTA, E.; LOPES, M.; FERNEDA, E. Mathema: A learning environment based on a multi-agent architecture. In: WAINER, J.; CARVALHO, A. **Advances in Artificial Intelligence**. Springer Berlin / Heidelberg, v. 991, 1995. p. 141-150.

9. DE BARROS COSTA, E.; PERKUSICH, A.; FERNEDA, E. **From a Tridimensional View of Domain Knowledge to Multi-agent Tutoring System**. 14th Brazilian Symposium on Artificial Intelligence. Springer Berlin / Heidelberg. 1998. p. 61-72.

10. SILVA, J. C.; DE BARROS COSTA, E.; FERNEDA, E. **Design and Development of an Authoring Environment for Building and Maintaining a Society of Artificial Tutoring Agents**. Simpósio Brasilieiro de Informática na Educação. Curitiba: SBC. 1999. p. 185-192.

11. DE ALMEIDA, H. O. et al. COMPOR: a Methodology, a Component Model, a Component based Framework and Tools to Build Multiagent Systems. **CLEI Electronic Journal**, v. 7, n. 1, p. 1-20, Junho 2004.

12. FRIGO, L. B.; CARDOSO, J.; BITTENCOURT, G. A method for modelling adaptive interactions in Intelligent Tutoring Systems. **International Journal of Continuing Engineering Education**, v. 17, n. 4/5, p. 381-391, 2007.

13. BITTENCOURT, I. B. et al. Sistemas de Autoria para Construção de Ambientes Interativos de Aprendizagem Baseada em Agentes. **Revista Brasileira de Informática na Educação**, v. 15, n. 1, p. 21-30, 2007.

14. COSTA, N. S. **Modelagem e Construção de uma Ferramenta de Autoria para um Sistema Tutorial Inteligente**. Universidade Federal de Maranhão. São Luís. 2002. Dissertação de Mestrado.

15. FRIGO, L. B.; CARDOSO, J.; BITTENCOURT, G. A Method for Modeling Adaptive Interactions in Intelligent Tutoring Systems. **International Journal of Continuing Engineering Education**, v. 17, p. 381-391, 2007.

16. SIBERTIN-BLANC, C. **High-level Petri nets with data structures**. European Workshop on Application and Theory of Petri Nets. Espoo, Finland: . 1985. p. 141-170.

17. HILL, E. J. F. **Jess, the Java Expert System Shell**. SANDIA National Laboratories. 2000.

18. CARDOSO, J. et al. **Petri Nets for Authoring Mechanism**. XV Simpósio Brasileiro de Informática na Educacão. Manaus: SBC. 2004. p. 378-387.

19. SILVA, J. C. **Aquisição de Conhecimento e Manutenção para uma Sociedade de Agentes Tutores**. Universidade Federal da Paraíiba. Campina Grande. 1999. Dissertação de Mestrado.

20. DE ALMEIDA, H. O. **Compor - Desenvolvimento de Software para Sistemas Multiagentes**. Universidade Federal de Campina Grande. Campina Grande. 2004. Dissertação de Mestrado.

21. BITTENCOURT, I. I. **Plataforma para Construçãoo de Ambientes Interativos de Aprendizagem baseados em Agentes**. Universidade Federal de Alagoas. Maceió. 2006. Dissertação de Mestrado.

22. BITTENCOURT, I. et al. **Um arcabouço para construção de sistemas baseados em agentes inteligentes**. XX Semana Paraense de Informática. 2006. p. .

23. BITTENCOURT, I. et al. **Ontologia para construção de ambientes interativos de aprendizagem**. XVII Simpósio Brasileiro de Informática na Educação. Brasília: SBC. 2006. p. 559-568.

24. BELLIFEMINE, F. L.; CAIRE, G.; GREENWOOD, D. **Developing Multi-Agent Systems with JADE**. 1st. ed. Wiley, 2007.

25. FRIGO, L. B. **Um Modelo para Autoria de Sistemas Tutores Adaptativos**. Universidade Federal de Santa Catarina. Santa Catarina. 2007. Tese de Doutorado

26. JENSEN, K.; KRISTENSEN, L. M.; WELLS, L. Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems. **STTT**, v. 9, n. 3-4, p. 213-254, 2007.

27. THE Description Logic Handbook: Theory, Implementation, and Applications. Description Logic Handbook. Cambridge University Press. 2003.

28. MURATA, T. Petri nets: Properties, analysis and applications. **Proceedings of the IEEE**, v. 77, n. 4, p. 541-580, Abril 1989.

29. WESTERGAARD, M. E.; KRISTENS, L. M. The Access/CPN Framework: A Tool for Interacting with the CPN Tools Simulator. In: _____ **Applications and Theory of Petri Nets**. Springer Berlin / Heidelberg, 2009. p. 313-322.

30. RUSSELL, S.; NORVIG, P. **Artificial Intelligence:** A Modern Approach. 3rd. ed. Prentice-Hall, 2009.

31. GAMMA, E. et al. **Design patterns:** elements of reusable object-oriented software. Addison-Wesley Professional, 1995.

32. SILVA, M. T.; BITTENCOURT, I. I.; COSTA, E. **Uma Abordagem Formal para Modelagem de Sistemas Multiagentes Utilizando Redes de Petri e Ontologias**. XXI Simpósio Brasileiro de Informática na Educação. 2010.

33. SILVA, M. T.; BITTENCOURT, I. I.; DE BARROS COSTA, E. Modelos para a Construção de Sistemas Multiagentes: Um Estudo de Caso em Sistemas Tutores Inteligentes. **Revista Brasileira de Informática na Educação**, v. 19, n. 1, p. 74-84, Agosto 2011.