Programas Baseados em Conhecimento: Uma Abordagem Inicial

Seiji Isotani[1]

[1]Aluno de Mestrado em Ciência da Computação
Instituto de Matemática e Estatística – Universidade de São Paulo (USP)

isotani@ime.usp.br

Resumo. *No desenvolvimento de sistemas computacionais complexos ou distribuídos, muitas vezes, gostaríamos de abstrair alguns detalhes de implementação e restrições do sistema, além de descrever o relacionamento entre o estado de "conhecimento" dos programas envolvidos e suas respectivas ações. Nesse contexto, este trabalho tem o intuito de introduzir as noções iniciais que facilitam o desenvolvimento de programas em termos de conhecimento, os chamados programas baseados em conhecimento (KBP), mostrar alguns aspectos e relações entre ações, protocolos e contextos no desenvolvimento dos KBP's e utilizar o problema das crianças enlameadas para firmar este conceito.*

1. Introdução

Como apresentado por [1], raciocinar sobre atividades em sistemas distribuídos no nível de conhecimento nos oferece algumas ferramentas para que possamos abstrair muitos dos detalhes de implementação do sistema que estamos considerando. Dessa forma, podemos intuitivamente pensar em maneiras mais abstratas de resolver problemas, focando principalmente nos conceitos essenciais e na descrição de um protocolo de alto-nível que possivelmente resolve o problema. Com este protocolo podemos fazer uma tradução para um programa concreto baseado nas propriedades particulares do sistema considerado.

A maneira como "atacamos" o problema acima é mais conhecida no mundo acadêmico como programação T*op-Down*. Esse tipo de desenvolvimento nos permite analisar e modificar programas mais facilmente, principalmente se considerarmos que um programa pode ser executado em diversos lugares com diferentes propriedades, como por exemplo, a confiabilidade da comunicação e de problemas em componentes do sistema.

O desenvolvimento de uma semântica formal para tornar possível a abordagem de desenvolvimento de programas como citados nos parágrafos anteriores motivou o surgimento da noção de *programas baseados em conhecimento* apresentado em [2][1], no qual as ações dos agentes dependem explicitamente de seus conhecimentos.

Para entender melhor vamos analisar o problema da transmissão de um bit apresentado em [5]. Imagine que temos dois processos, o Emissor (S) e o Receptor (R) que se comunicam através de uma linha de comunicação. Podemos resolver o problema de duas maneiras como mostra a tabela 1a e 1b. A primeira forma (tabela 1a) é conhecida como um programa normal, a segunda forma (tabela 1b) é conhecida como um

programa baseado em conhecimento, pois o teste if $\neg \mathbf{K}_S \mathbf{K}_R(\mathbf{bit})$ é conhecido como um teste de conhecimento e pode ser lido como "o Emissor não sabe que o Receptor sabe bit".

S : if **¬recack** do sendbit R : if **recbit** do sendack	S : if $\neg \mathbf{K}_S \mathbf{K}_R(\mathbf{bit})$ do sendbit R : if $\mathbf{K}_R(\mathbf{bit}) \wedge \neg \mathbf{K}_R \mathbf{K}_s \mathbf{K}_R(\mathbf{bit})$ do sendack

Tabela 1a. Programa Normal (Standard).

Tabela 1b. Programa baseado em conhecimento (KBP).

Podemos observar que a primeira forma de resolver o problema faz com que R fique enviando a confirmação de recebimento do bit após saber qual bit foi enviado por S. Ao contrario da segunda forma que faz R parar de enviar a confirmação após saber que S sabe que ele sabe o valor do bit. Além disso, a segunda forma abstrair a maneira na qual S identifica que R sabe o valor do bit e a maneira na qual R identifica que S sabe que R sabe o valor do bit.

Um programa baseado em conhecimento pode ser visto com uma especificação em alto nível, e portanto, podem existir várias ou nenhuma implementação que satisfaça esta especificação. Isso se deve ao fato da circularidade da definição de programas baseados em conhecimento: as ações são executadas dependendo do conhecimento dos agentes, mas o conhecimento dos agentes dependem das ações que foram executadas[3].

Para formalizar a descrição dos programas baseados em conhecimento o presente trabalho irá apresentar alguns conceitos iniciais necessários para o entendimento do texto. A seção 2 trata destes conceitos cujo conteúdo são as definições de ações, protocolos, contexto e consistência. Na seção 3 vamos definir o que são os programas normais (Standards) e finalmente na seção 4 definiremos os programas baseados em conhecimento(KBP). A seção 5 é destinada a desenvolver o problema das crianças enlameadas e por último na seção 6 teremos as conclusões deste trabalho.

2. Definições Iniciais

Nós queremos viabilizar a construção de uma semântica que possa descrever a interação entre agentes, ou seja, processos executando um protocolo, em um determinado contexto, fazendo algum tipo de ação. Para isso, iremos fazer uso das estruturas apresentadas em [5] e faremos um reforço apresentando uma revisão detalhada de algumas das principais estruturas que serão utilizadas para a definição dos KBP's.

2.1. Ações

A definição de ações para nós segue da seguinte frase retirada de [1]: as ações são simplesmente elementos de um conjunto que causam uma mudança no estado do sistema. Podemos imaginar como uma execução *r* de um sistema surge, e intuitivamente, deduzir que as execuções surgem a partir de mudanças de estados no sistema que ocorrem como resultados das ações de agentes e do ambiente.

Podemos assumir que cada agente *i* contém um conjunto de ações, denotado por $\mathbf{ACT}_i$, que podem ser executadas por *i*. Por exemplo, no problema da transmissão de um bit apresentado na seção anterior, uma ação executada pelo agente Emissor (S) seria a passagem da mensagem *sendbit* para o agente Receptor (R). Mantendo este mesmo

ponto de vista, o ambiente pode ser tomando como um agente, cujo estado de conhecimento não é de nosso interesse, contendo um conjunto de ações $\mathbf{ACT_e}$ que podem ser executadas pelo ambiente. Novamente utilizando o problema da transmissão de um bit, podemos imaginar o ambiente como um agente que envia ou não as mensagens do Emissor para o Receptor ou vice-versa. Para ambos, agente e ambiente, é permitido a execução de uma ação especial denominada ação nula Λ, ou seja, para o sistema seria como se nenhuma ação fosse executada pelo agente ou pelo ambiente. Um exemplo de conjunto de ações para o problema da transmissão de um bit pode ser visto na tabela 2.

$\mathbf{ACT_S} = \{\text{sendbit}, \Lambda\}$
$\mathbf{ACT_R} = \{\text{sendack}, \Lambda\}$
$\mathbf{ACT_e} = \{(a, b) \mid a \in \{\text{delivers}, \Lambda\}, b \in \{\text{delivers}, \Lambda\}\}$

Tabela 2. Conjunto de ações para o problema da transmissão de um bit.

Sabemos que ações executadas por um agente em particular não são suficientes para determinar as mudanças do estado global do sistema [1]. Além disso, ações simultâneas executadas por diferentes agentes podem interagir, como por exemplo, se dois agentes tentarem escrever ao mesmo tempo um valor no banco de dados nós não temos uma noção clara do que vai acontecer. Para lidar com essa problema nos iremos considerar uma ação conjunta que será denotada pela tupla $(\mathbf{a_e}, \mathbf{a_1}, \ldots, \mathbf{a_n})$, onde $\mathbf{a_e}$ é a ação executada pelo ambiente e $\mathbf{a_i}$ é a ação executada pelo agente *i*, para *i* = 1, ..., n.

Para que a ação conjunta modifique o estado do sistema vamos definir uma função τ, chamada função de transição, cujo objetivo é associar a ação conjunta $(\mathbf{a_e}, \mathbf{a_1}, \ldots, \mathbf{a_n})$ à um estado global do sistema $(\ell_e, \ell_1, \ldots, \ell_n)$. O resultado desta função é um outro estado global do sistema que pode ser igual ou diferente do anterior. Assim, a definição de função de transição pode ser vista como:

Função de Transição: $\tau : \text{ACT X G} \Rightarrow \text{G}$. Ou seja,
$\tau(\mathbf{a_e}, \mathbf{a_1}, \ldots, \mathbf{a_n})(\ell_e, \ell_1, \ldots, \ell_n) = (\ell'_e, \ell'_1, \ldots, \ell'_n)$

2.2. Protocolos

No tópico anterior definimos o que são as ações dos agentes e como aplica-las a um estado global do sistema, porém uma dúvida surgiria imediatamente após essa definição: Quais ações um agente *i* pode executar dado seu estado local?

A resposta para a pergunta acima é a definição de protocolos. Intuitivamente, um protocolo para um agente *i* é a descrição de quais ações o agente *i* pode realizar dado seu estado local. Formalmente, um protocolo $\mathbf{P_i}$ é definido como uma função de um conjunto $\mathbf{L_i}$ de estados locais de um agente *i* para um sub-conjunto não vazio de ações em $\mathbf{ACT_i}$. Esta formalização permite a existência de protocolos não-determinísticos, ou seja, $P_i(\ell_i) = \{a_1, a_2\}$, onde $\ell_i \in L_i$ e $\{a_1, a_2\} \in ACT_i$. Um protocolo é determinístico se $P_i(\ell_i) = \{a\}$ para cada estado local $\ell_i \in L_i$.

Da mesma forma como em agentes, as ações executadas pelo ambiente podem ser formalizadas utilizando a noção de protocolo. Assim, um protocolo $\mathbf{P_e}$ para o ambiente

é definido como uma função de um conjunto $\mathbf{L_e}$ para um sub-conjunto não vazio de ações em $\mathbf{ACT_e}$. Note que, se todos os agentes e o ambiente seguirem protocolos determinísticos, então existe apenas uma linha de execução para um determinado estado global inicial.

Observe que a evolução do sistema é causada pela combinação de protocolos executados pelos agentes e pelo ambiente. Portanto, podemos definir a idéia de protocolo conjunto $\mathbf{P = (P_1, ... ,P_n)}$, onde $\mathbf{P_i}$ é o protocolo do agente *i*, para *i* = 1, ..., n. Como podemos observar, o protocolo do ambiente não está incluso em **P**. Isso se deve ao fato que o ambiente é considerado um caso especial,e muitas vezes, o ambiente é visto como uma adversidade que pode fazer com que o sistema se comporte de maneira indesejada.

Resumindo o que vimos aqui:

> **Definição:** Um protocolo P_i para um agente i é o mapeamento de um conjunto L_i de estados locais dos agentes i para um conjunto não vazio de ações em ACT_i. O mesmo vale para o protocolo do ambiente P_e. [1]

2.3. Contexto

Os protocolos apresentados no tópico anterior descrevem apenas as ações possíveis de serem executadas pelos agentes ou pelo ambiente. Mas, para determinar o comportamento do sistema será necessário mais um elemento, o chamado *contexto*. O contexto no qual os protocolos são executados é que determinam o comportamento do sistema.

Podemos tentar identificar quais são os elementos que pertencem ao contexto. Em primeiro lugar, como o protocolo do ambiente $\mathbf{P_e}$ não pertence ao protocolo conjunto **P**, é claro que este deve fazer parte do contexto, garantindo a contribuição das ações do ambiente no sistema. Além disso, no contexto deve ser incluído a função de transição $\boldsymbol{\tau}$, pois esta descreve os resultados das ações aplicadas num determinado estado global do sistema. Observe que incluindo $\boldsymbol{\tau}$, implicitamente estamos incluindo os conjuntos $\mathbf{L_e, L_1, ..., L_n}$ de estados locais e também, os conjuntos $\mathbf{ACT_e, ACT_1, ..., ACT_n}$ de ações, pois $\boldsymbol{\tau}$ tem como domínio estes conjuntos. Finalmente, no contexto deve ser incluído um conjunto $\mathbf{G_0}$ de estados globais iniciais, pois este descreve o estado do sistema quando a execução do protocolo é iniciada.

Muitas vezes queremos considerar mais algumas restrições globais para definir o comportamento do sistema que não podem ser capturadas por $\mathbf{P_e}$, $\boldsymbol{\tau}$ e $\mathbf{G_0}$, como por exemplo: "todas as mensagem são em algum momento entregues"[1]. Para permitir esse tipo de restrição admitimos no contexto a existência de $\boldsymbol{\Psi}$. Formalmente, $\boldsymbol{\Psi}$ é um conjunto de execuções, sendo que uma execução $\mathbf{r} \in \boldsymbol{\Psi}$ sse **r** satisfaz a condição $\boldsymbol{\Psi}$.

Resumindo o que vimos aqui:

> **Definição:** Um contexto $\boldsymbol{\gamma}$ é uma tupla $(\mathbf{P_e, G_0, \tau, \Psi})$, onde $P_e : L_e \rightarrow 2^{ACT_e} - \{\varnothing\}$ é um protocolo para o ambiente, $\mathbf{G_0}$ é um subconjunto não vazio do conjunto G de estados globais, $\boldsymbol{\tau}$ é a função de transição e $\boldsymbol{\Psi}$ é uma condição de admissibilidade numa execução. [1]

2.3. Consistência e Sistema

Após as definições apresentadas acima nós podemos falar um pouco sobre execuções de um protocolo num determinado contexto. O que queremos aqui é definir o que é uma execução consistente e um sistema que representa um protocolo.

Em primeiro lugar iremos definir consistência[1]:

Definição: Uma execução *r* é fracamente consistente com um protocolo conjunto $\mathbf{P} = (\mathbf{P_1}, \ldots, \mathbf{P_n})$ no contexto $\gamma = (\mathbf{P_e}, \mathbf{G_0}, \tau, \Psi)$ se:

1. $r(0) \in \mathbf{G_0}$
2. Para todo $m \geq 0$, se $r(m) = (\ell_e, \ell_1, \ldots, \ell_n)$, então existe uma ação conjunta $(a_e, a_1, \ldots, a_n) \in P_e(\ell_e) \times P_1(\ell_1) \ldots P_n(\ell_n)$ tal que $r(m+1) = \tau(a_e, a_1, \ldots, a_n)(r(m))$.

A execução é consistente com **P** no contexto γ se é fracamente consistente e satisfaz:

3. $r \in \Psi$

Podemos então dizer que uma execução é consistente com **P** no contexto γ se r é um possível comportamento do sistema através das ações descritas por **P**.

Agora podemos definir Sistema[1]:

Definição: Um sistema representa o protocolo **P** no contexto γ, denotado por $\mathbf{R}^{rep}(\mathbf{P}, \gamma)$, é o sistema que consiste de todas as execuções consistentes com **P** no contexto γ.

3. Programa Normal (Stardard)

Para descrever programas normais, apresentaremos uma linguagem simples, porém suficiente para descrever-los e cuja sintaxe dá ênfase no fato que a execução do agente está baseada nos testes aplicados em seu estado local. Assim um programa normal $\mathbf{Pg_i}$ para um agente *i* é apresentado na tabela 3 [1][3], onde os $\mathbf{t_j}$'s são testes normais para o agente *i* e os $\mathbf{a_j}$'s são as ações possíveis para o agente *i*.

```
case of
    if t1 do a1
    if t2 do a2
      ...
end case
```

Tabela 3. Sintaxe para um Programa Normal.

Um teste normal para um agente i é simplesmente uma formula proposicional do conjunto Φ_i de proposições primitivas[3], cuja valoração irá depender do estado local do agente *i* (observe que todas as proposições primitivas em Φ_i devem ser locais para o agente *i*).

Uma vez que saibamos como avaliar os testes a partir dos estados locais $\mathbf{L_i}$ do agente *i*, podemos então converter o programa para um protocolo em $\mathbf{L_i}$, ou seja, num determinado estado local $\ell \in L_i$, o agente *i* irá executar as ações cujo testes em ℓ sejam verdadeiros.

Para resolver o problema acima, ou seja, saber como avaliar os testes, nos faremos uso de uma interpretação π. Esta interpretação num determinado estado global **G** é dita *compatível* com um programa $\mathbf{Pg_i}$ para um agente *i* se cada proposição que aparece em $\mathbf{Pg_i}$ é local para *i*.

Exemplificando o parágrafo anterior, se φ é uma formula proposicional, então todas as formulas primitivas que pertencem a φ são locais para o agente *i* e se ℓ for seu estado local, denotamos $(\pi,\ell) \models \varphi$ se φ satisfaz $\pi(\mathbf{g})(\varphi) = \mathbf{true}$, $\mathbf{g} = (\ell_e, \ell_1, \ldots, \ell_n)$ e $\ell_i = \ell$.

Dado um programa $\mathbf{Pg_i}$ para um agente *i* e uma interpretação π compatível com $\mathbf{Pg_i}$, podemos definir um protocolo que será denotado por $\mathbf{Pg_i^{\pi}}$ [1][3].

Definição:

$$\mathbf{Pg_i^{\pi}}(\ell) = \begin{cases} \{\mathbf{a_j} : (\pi,\ell) \models \mathbf{t_j}\} & \text{if } \{\mathbf{j} : (\pi,\ell) \models \mathbf{t_j}\} \neq \varnothing \\ \{\Lambda\} & \text{if } \{\mathbf{j} : (\pi,\ell) \models \mathbf{t_j}\} = \varnothing \end{cases}$$

$$\textbf{Onde } (\pi,\ell) \models \varphi \text{ se } \varphi \text{ satisfaz } \pi(\mathbf{g})(\ell) = \mathbf{true},$$
$$\mathbf{g} = (\ell_e, \ell_1, \ldots, \ell_n) \text{ e } \ell_i = \ell$$

Para finalizar, vamos analisar as definições apresentadas nesta seção em conjunto com as seções anteriores reforçando assim, os conceitos vistos até o momento.

- Um programa conjunto é uma tupla $\mathbf{Pg} = (\mathbf{Pg_1}, \ldots, \mathbf{Pg_n})$, onde $\mathbf{Pg_i}$ é um programa normal para o agente *i*.
- Dada uma interpretação π compatível com cada $\mathbf{Pg_i}$, então um protocolo conjunto é definido como $\mathbf{Pg^{\pi}} = (\mathbf{Pg_1^{\pi}}, \ldots, \mathbf{Pg_n^{\pi}})$. (compatibilidade foi definida alguns parágrafos acima).
- Um contexto interpretado é um par (γ,π), que consiste de um contexto γ e uma interpretação π.
- Um sistema interpretado $I = (\mathcal{R},\pi)$ representa o programa conjunto **Pg** no contexto interpretado (γ,π) se π é compatível com **Pg** e $\mathcal{R}$ representa o protocolo correspondente $\mathbf{Pg^{\pi}}$ no contexto γ. Esse sistema será denotado por $\mathbf{I}^{rep}(\mathbf{Pg},\gamma,\pi)$.

4. Programa Baseado em Conhecimento (KBP)

A seção anterior apresentou a noção de um programa normal (standard) onde uma agente seleciona suas ações com base nos resultados de testes que dependem apenas de seu estado local. Porém, esta noção não pode ser usada para descrever relações entre o conhecimento dos agentes e suas ações como gostaríamos.

Utilizando a mesma linguagem utilizada para descrever os programas normais apresentados na seção 3, iremos estende-la para formalizar a descrição de um *programa baseado em conhecimento* $\mathbf{Pg_i}$ para um agente *i* como mostra a tabela 4 [1][3], sendo que os $\mathbf{t_j}$'s são testes normais para *i*, os $\mathbf{k_j}$'s são testes de conhecimento para *i* e os $\mathbf{a_j}$'s são as ações possíveis para *i*.

case of
  if $\mathbf{t_1} \wedge \mathbf{k_1}$ do $\mathbf{a_1}$
  if $\mathbf{t_2} \wedge \mathbf{k_2}$ do $\mathbf{a_2}$
  ...
end case

Tabela 4. Sintaxe para um Programa Baseado em conhecimento.

Um teste de conhecimento não pode ser determinado olhando apenas os estados locais dos agentes isoladamente. O teste de conhecimento para um agente *i* é uma combinação booleana de formulas da forma $\mathbf{k_i}\varphi$, onde φ pode ser uma formula qualquer podendo incluir outros operadores como o de conhecimento comum e operadores temporais.[1]

Para efeito de exemplo, vamos considerar o problema das crianças enlameadas (este problema será discutido mais na seção 5). Neste problema existe um número **k** de crianças com as testas sujas de lama e seu pai diz "pelo menos uma de vocês está com a testa suja". Após esta afirmação, ele repetidamente pergunta para as crianças se elas sabem se estão com a testa suja. Caso saibam a resposta, as crianças dizem "sim", caso contrário, dizem "não". Se a proposição $\mathbf{p_i}$ representa "a criança i está com a testa suja", então podemos escrever um programa da seguinte forma:

case of
  if $\mathbf{childheard_i} \wedge (\mathbf{k_i p_i} \vee \mathbf{k_i} \neg \mathbf{p_i})$ do say "Sim"
  if $\mathbf{childheard_i} \wedge \neg \mathbf{k_i p_i} \vee \neg \mathbf{k_i} \neg \mathbf{p_i}$ do say "Não"
end case

A proposição $\mathbf{childheard_i}$ é considerada verdadeira se a criança *i* escutou o que o pai disse. Intuitivamente, a valoração como verdadeira desta proposição só depende do próprio estado da criança *i*. Ou seja este teste é considerado um teste normal como definido na seção 3. Os testes $(\mathbf{k_i p_i} \vee \mathbf{k_i} \neg \mathbf{p_i})$ e $\neg \mathbf{k_i p_i} \vee \neg \mathbf{k_i} \neg \mathbf{p_i}$ são considerados testes de conhecimento, e portanto, não podem ter sua valoração determinada olhada apenas o estado local da criança *i*.

Em programas normais, nos deduzimos os protocolos a partir da descrição dos programas normais e de uma interpretação π. O que queremos agora é a partir da descrição de programas baseados em conhecimento produzir protocolos para estes. Porém esta tarefa não é fácil já que em programas baseados em conhecimento precisamos avaliar testes de conhecimento que dependem de todo o sistema interpretado. Isso acontece pois em um mesmo local ℓ em dois diferentes sistemas interpretados I_1 e I_2, o teste $\mathbf{k_i}\varphi$ pode ser verdadeiro no estado local ℓ em I_1 e falso no estado local ℓ em I_2.

Para resolver este problema, nos associamos aos programas baseados em conhecimento um sistema interpretado , assim podemos denotar um protocolo para um agente *i* como $\mathbf{Pg}_i^I$. Intuitivamente iremos avaliar os testes normais em $\mathbf{Pg}_i$ de acordo com π, levando em conta todas as considerações definidas na seção 3, e avaliar os testes de conhecimento em $\mathbf{Pg}_i$ de acordo com I.

Para avaliar o teste de uma proposição φ num determinando estado local do agente *i* em um sistema interpretado I, denotado por $(I,\ell) \models \varphi$, faremos o seguinte:

- Se φ é um teste normal e $I = (\mathcal{R}, \pi)$ então, como apresentado na seção 3, nos definimos $(I,\ell) \models \varphi$ sse $(\pi,\ell) \models \varphi$. Como φ é um teste normal em $\mathbf{Pg}_i$, então as proposições de φ devem ser locais, e portanto, esta definição faz sentido já que pela definição os programas normais podem ser considerados um caso particular dos KBP's.
- Se φ é um teste de conhecimento da forma $\mathbf{k}_i\psi$ então definimos $(I,\ell) \models \mathbf{k}_i\psi$ sse $(I,\mathbf{r},\mathbf{m}) \models \psi$ para todos os pontos $(\mathbf{r},\mathbf{m})$ de I tal que $\mathbf{r}_i(\mathbf{m}) = \ell$. Observe que esta definição de conhecimento é basicamente igual a definida em [5], $(I,\mathbf{r},\mathbf{m}) \models \mathbf{k}_i\psi$ sse $(I,\mathbf{r}',\mathbf{m}') \models \psi$ para todo $(\mathbf{r}',\mathbf{m}')$ tal que $(\mathbf{r},\mathbf{m}) \sim_i (\mathbf{r}',\mathbf{m}')$, porém com uma condição a mais $\mathbf{r}_i(\mathbf{m}) = \ell$. Para exemplificar, vamos olhar as figuras 1a e 1b. O primeiro deles é a definição dada em [5] e o seguindo é a definição apresentada neste parágrafo.

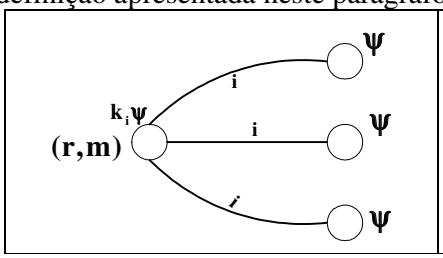

Figura 1a.

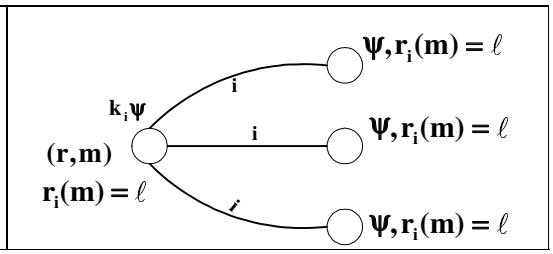

Figura 1b.

Dado um programa $\mathbf{Pg}_i$ para um agente i e um sistema interpretado $I = (\mathcal{R}, \pi)$ definimos um protocolo $\mathbf{Pg}_i^I$ da seguinte forma [1][3]:

Definição:

$$\mathbf{Pg}_i^I(\ell) = \begin{cases} \{\mathbf{a}_j : (I,\ell) \models \mathbf{t}_j\} & \text{if } \{\mathbf{j} : (I,\ell) \models \mathbf{t}_j\} \neq \varnothing \\ \{\Lambda\} & \text{if } \{\mathbf{j} : (I,\ell) \models \mathbf{t}_j\} = \varnothing \end{cases}$$

Onde $(I,\ell) \models \varphi$ sse $(\pi,\ell) \models \varphi$;
$(I,\ell) \models \mathbf{k}_i\psi$ sse $(I,\mathbf{r},\mathbf{m}) \models \psi$ para todos os pontos
(r,m) de I tal que $\mathbf{r}_i(\mathbf{m}) = \ell$.

Finalmente para finalizar esta seção, iremos analisar as definições apresentadas nesta aqui em conjunto com as seções anteriores como fizemos na seção 3. Assim:

- Um KBP conjunto é uma tupla $\mathbf{Pg} = (\mathbf{Pg_1}, \ldots, \mathbf{Pg_n})$, onde $\mathbf{Pg_i}$ é um programa baseado em conhecimento para o agente *i*.
- Dado um sistema interpretado $I = (\mathcal{R}, \pi)$, um protocolo conjunto é definido como $\mathbf{Pg}^I = (\mathbf{Pg}_1^I, \ldots, \mathbf{Pg}_n^I)$.
- Um sistema interpretado $I = (\mathcal{R}, \pi)$ representa o programa conjunto **Pg** no contexto interpretado (γ, π) se π é compatível com **Pg** e $\mathcal{R}$ representa o protocolo correspondente $\mathbf{Pg}^I$ no contexto γ. Observe que se I representa o protocolo obtido pela avaliação de testes de conhecimento no programa **Pg** com respeito à si próprio, e por causa desta circularidade na definição podem existir nenhum, um ou vários sistemas interpretados que representem este KBP.
- Como um KBP não pode ser implementado diretamente [2], podemos então tentar definir a implementação de um KBP a partir de um programa normal. Ou seja: Um programa normal $\mathbf{Pg_s}$ é uma implementação de um KBP $\mathbf{Pg_{kb}}$ no contexto interpretado (γ, π) se o protocolo $\mathbf{Pg}^\pi$ implementa $\mathbf{Pg_{kb}}$ em (γ, π). $\mathbf{Pg}^\pi$ implementar $\mathbf{Pg_{kb}}$ significa que $\mathbf{Pg}^\pi$ e $\mathbf{Pg}_i^I$ tem os mesmos comportamentos em todos os estados globais que aparecem em I. Observe que da mesma forma que pode existir nenhum ou vários sistemas interpretados que representem o KBP, pode existir nenhum ou vários programas normais que podem implementar um KBP.

5. Um Exemplo de aplicação: O problema das Crianças Enlameadas

Conhecido como Muddy Children Puzzle, este problema consiste no seguinte:

> Imagine que **n** crianças estão brincando juntas. A mãe destas crianças avisou-as que se elas voltassem sujas para casa iriam receber um castigo. Então cada criança tentará se manter limpa, porém elas adoram ver as outras crianças sujas. Durante a brincadeira **k** crianças sujaram suas testas. Cada uma pode ver se há sujeira na testa da outra, porém não pode ver a sua própria testa. Quando o pai chega para busca-las, ele diz: "pelo menos uma de vocês está com a testa suja". Após a afirmação o pai pergunta diversas vezes: "Vocês sabem se estão com a testa suja?". Assumindo que as crianças são perceptivas, inteligentes, só dizem a verdade e respondem simultaneamente, podemos mostrar que na k-ésima vez que o pai fizer a pergunta, as crianças com a testa suja irão responder "Sim".

Supondo um sistema síncrono, ou seja, todas as crianças respondem ao mesmo tempo, vamos analisar como as crianças chegam a conclusão que estão com a testa suja. Seja k=1, o resultado é simples: a criança com a testa suja vê que não existe outra criança com a testa seja. Como ela sabe que pelo menos uma criança está com a testa suja, ela conclui que ela deve estar com a testa suja e responde "Sim". Supondo agora que k=2, então existem 2 crianças sujas, digamos, *a* e *b*. Nenhuma delas responde a pergunta, ou seja, não sabem se estão com a testa suja, na primeira vez que o pai pergunta. Isso ocorre porque uma vê a testa suja da outra. Quando *b* não responde, *a* percebe que sua testa deve estar suja, pois caso contrário *b* deveria saber que sua testa estava suja e teria respondido "Sim" na primeira vez que o pai perguntará. Portanto *a* responde "Sim" na

próxima vez que o pai perguntar. O mesmo raciocínio fará *b* que também responderá "Sim" na próxima vez. Agora vamos supor que k=3, então existem 3 crianças sujas, *a*, *b* e *c*. A criança *a* pensa da seguinte forma: Suponha que eu não esteja com a testa suja, então se k=2, *b* e *c* responderão "Sim" na segunda vez que o pai perguntar. Como isso não acontece então ele percebe que sua suposição era falsa, e portanto, sabe que está com a testa suja respondendo "Sim" quando o pai perguntar pela tercei ra vez. O mesmo raciocínio vale para *b* e *c*.

Como apresentado em [4], a maneira como a criança *i* pensa segue a seguinte formula:

$$\mathbf{B_i}(\neg \mathbf{p_i} \wedge \mathbf{B_1}(\neg \mathbf{p_1} \wedge \mathbf{B_2}(\neg \mathbf{p_2} \wedge ... \mathbf{B_{k-1}}(\neg \mathbf{p_{k-1}} \wedge (\mathbf{k_k p_k}))...)))$$

Onde $\mathbf{p_i}$ significa que a criança *i* está com a testa suja e $\mathbf{B_i}$ pode ser lido como "a criança *i* acredita que ...". Assim, a formula para uma criança *i* pode ser entendida da seguinte forma (k>1): Suponha que eu não esteja com a testa suja, então deve existir k-1 crianças com a testa suja, assim a criança 1 deve estar vendo k-2 crianças sujas já que não pode ver se sua testa está suja, e deve supor como eu que sua testa não está suja, assim ela acredita que a criança 2 está vendo k-3 crianças com a testa suja .... Esse raciocínio vai até chegarmos num estado onde se acredita que a criança *k* sabe que está com a testa suja.Quando nenhuma criança responde a pergunta do pai então a criança *i* percebe que sua suposição era falsa mudando-a para a formula abaixo:

$$\mathbf{B_i}(\neg \mathbf{p_i} \wedge \mathbf{B_1}(\neg \mathbf{p_1} \wedge \mathbf{B_2}(\neg \mathbf{p_2} \wedge ... \mathbf{k_{k-1}}(\mathbf{p_{k-1}} \wedge (\mathbf{k_k p_k}))...)))$$

Para escrever o KBP para este problema precisamos de mais algumas proposições:

- $\mathbf{inicial_i}$ - sendo verdadeira quando a criança *i* está em seu estado inicial.
- **add** – adiciona um conhecimento na base de conhecimento do agente.

Assim um programa baseado em conhecimento para resolver este problema pode ser visto na tabela 5. Note que $\mathbf{k_i}(\bigvee_{j=1,\, j\neq i}^{n} \mathbf{k_j p_j})$, significa que o agente *i* sabe que existe pelo menos uma criança *j* que sabe que está com a testa suja, ou seja, disse "Sim", então *i* sabe que ele não está com a testa suja, pois seguem o mesmo raciocínio acima.

case of
  if $\mathbf{inicial_i} \wedge \mathbf{k_i p_i}$ then say "Sim"
  if $\neg \mathbf{k_i p_i} \wedge \mathbf{k_i}(\bigvee_{j=1,\, j\neq i}^{n} \mathbf{k_j p_j})$ then $\mathbf{add}\ \mathbf{k_i} \neg \mathbf{p_i}$
  if $\mathbf{B_i}(\neg \mathbf{p_i} \wedge \mathbf{B_1}(\neg \mathbf{p_1} \wedge \mathbf{B_2}(\neg \mathbf{p_2} \wedge ... \mathbf{B_{k-1}}(\neg \mathbf{p_{k-1}} \wedge (\mathbf{k_k p_k}))...)))$ then
    $\mathbf{add}\ \mathbf{B_i}(\neg \mathbf{p_i} \wedge \mathbf{B_1}(\neg \mathbf{p_1} \wedge \mathbf{B_2}(\neg \mathbf{p_2} \wedge ... \mathbf{k_{k-1}}(\mathbf{p_{k-1}} \wedge (\mathbf{k_k p_k}))...)))$
    if $\mathbf{k_i p_i}$ then say "Sim"
end case

Tabela 5. KBP para o problema das crianças enlameadas sincrono.

Vamos tentar analisar o problema apresentado nesta seção utilizado a estrutura gráfica de Kripke para o problema das crianças enlameadas com n=3 e descobrir porque a formula apresentada em [4] funciona.

Antes do anúncio do pai que diz: "Existe pelo menos uma criança com a testa suja", a estrutura de Kripke para este problema é apresentada na figura 2a, e após este anúncio a estrutura fica na forma da figura 2b.

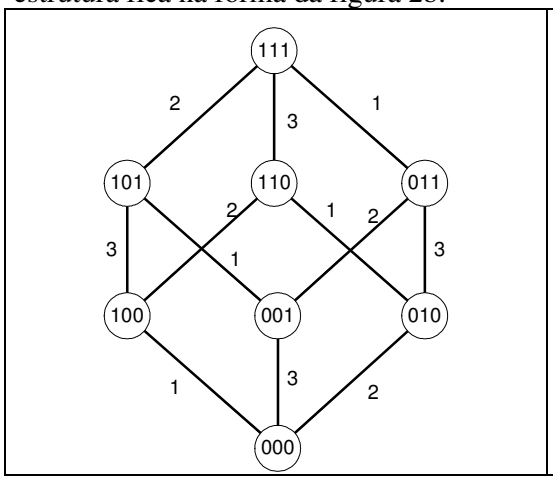

Figura 2a.

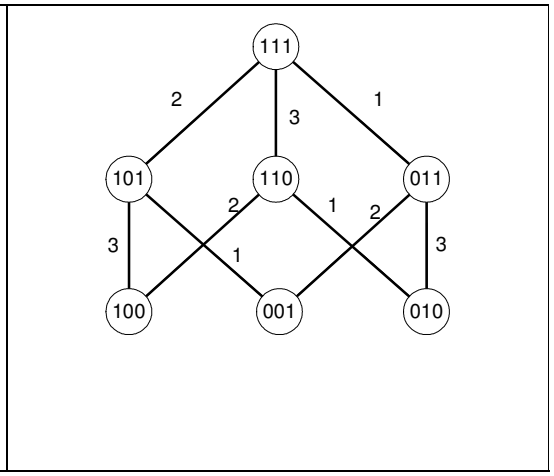

Figura 2b.

1) Vamos imaginar que k=2 e que as crianças 1 e 2 estão sujas. O conhecimento inicial para cada criança é: (1) criança 1 $\mathbf{k_1p_2, k_1\neg p_3}$, (2) criança 2 $\mathbf{k_2p_1, k_2\neg p_3}$ (3) criança 3 $\mathbf{k_3p_1, k_3p_2}$

Analisando a criança 1: a criança 1 vê que a 2 está com a testa suja e 3 esta com a testa limpa. A figura 3a mostra o estado da criança 1. Ela não pode diferenciar entre os estados (110) e (010). Como apresentado em [4], a criança sempre acredita que esta com a testa limpa, e portanto, como ela sabe que pelo menos uma criança está suja e 2 está suja então ela supõe que k=1, ou seja, esta no estado (010) e sua testa está limpa (figura 3b). Nesse estado apenas a criança 2 está com testa suja, e portanto, como ela sabe que existe pelo menos uma criança suja, então ela responderia "Sim" quando o pai fizesse a pergunta. Assim, a formula que descreve o raciocínio da criança 1 é: $\mathbf{B_1(\neg p_1 \wedge k_2p_2)}$.

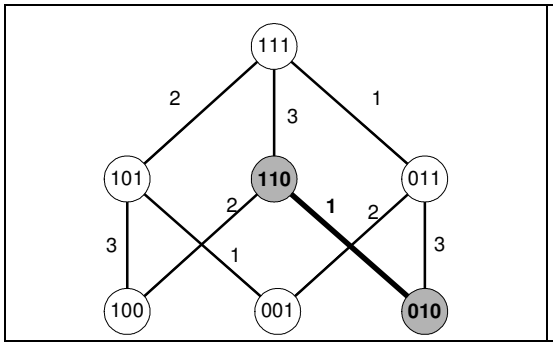

Figura 3a. A criança 1 não diferencia os estados (110) e (010).

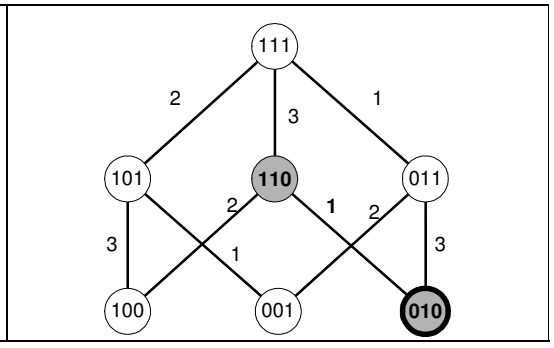

Figura 3b. A Criança 1 acredita que está com a testa limpa.

Quando a criança 2 não responde na primeira vez que o pai pergunta, 1 percebe que sua suposição não é verdade e então sua testa precisa estar suja e da mesma forma a criança 1 sabe que a criança 2 também sabe que está com a testa suja, pois fazem o mesmo

raciocínio, ou seja, $\mathbf{k_1(p_1 \wedge k_2 p_2)}$. Assim, uma execução do problema das crianças enlameadas síncrono com n=3 e k=2 é apresentado abaixo:

Criança 1	Criança 2	Criança 3
Conhecimento inicial: $\mathbf{k_1 p_2, k_1 \neg p_3}$	Conhecimento inicial: $\mathbf{k_2 p_1, k_2 \neg p_3}$	Conhecimento inicial: $\mathbf{k_3 p_1, k_3 p_2}$
Suposição inicial: $\mathbf{B_1(\neg p_1 \wedge k_2 p_2)}$	Suposição inicial: $\mathbf{B_2(\neg p_2 \wedge k_1 p_1)}$	Suposição inicial: $\mathbf{B_3(\neg p_3 \wedge B_1(\neg p_1 \wedge k_2 p_2))}$
O pai pergunta (1 vez)	O pai pergunta (1 vez)	O pai pergunta (1 vez)
Novo conhecimento $\mathbf{k_1(p_1 \wedge k_2 p_2)}$	Novo conhecimento: $\mathbf{k_2(p_2 \wedge k_1 p_1)}$	Nova suposição: $\mathbf{B_3(\neg p_3 \wedge k_1(p_1 \wedge k_2 p_2))}$
O pai pergunta (2 vez) Responde "Sim"	O pai pergunta (2 vez) Responde "Sim"	

2) Vamos imaginar que k=3, então as 3 crianças estão sujas.

Analisando a criança 1: a criança 1 vê que as crianças 2 e 3 estão com as testas sujas. A figura 4a mostra o estado da criança 1. Como no exemplo anterior, ela não pode diferenciar os estados (111) e (011). E novamente como apresentado em [4], a criança sempre acredita que está com a testa limpa e portanto, supõem que está no estado (011) e que k=2 (figura 4b).

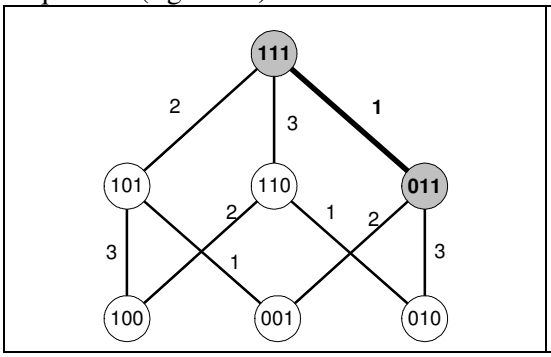

Figura 4a. A criança 1 não diferencia os estados (111) e (011).

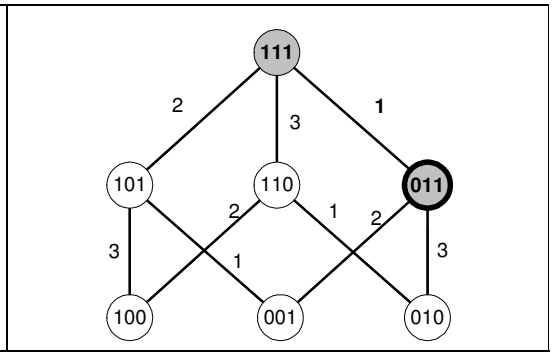

Figura 4b. A Criança 1 acredita que esta com a testa limpa.

Como a criança 1 sabe que as outras crianças pensam da mesma forma que ela, então se 1 acredita que está no estado (011), 1 supõem que a criança 2 raciocina neste estado e, assim como ela, não sabe se está com a testa limpa (figura 5a) e da mesma forma supõe que está com a testa limpa (figura 5b). A formula que descreve o raciocínio da criança 1 é: $\mathbf{B_1(\neg p_1 \wedge B_2(\neg p_2 \wedge k_3 p_3))}$.

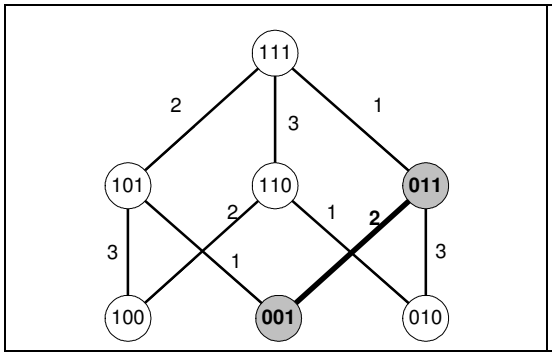

Figura 5a. A criança 2 não diferencia os estados (011) e (001).

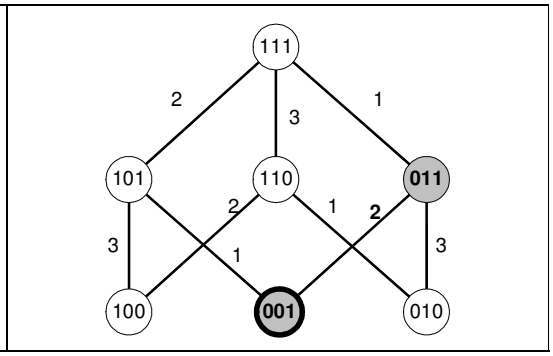

Figura 5b. A Criança 2 acredita que esta com a testa limpa.

Uma execução para o problema das crianças enlameadas com k=3 e n=3 é o seguinte:

Criança 1	Criança 2	Criança 3
Conhecimento inicial: $\mathbf{k}_1\mathbf{p}_2, \mathbf{k}_1\mathbf{p}_3$	Conhecimento inicial: $\mathbf{k}_2\mathbf{p}_1, \mathbf{k}_2\mathbf{p}_3$	Conhecimento inicial: $\mathbf{k}_3\mathbf{p}_1, \mathbf{k}_3\mathbf{p}_2$
Suposição inicial: $\mathbf{B}_1(\neg\mathbf{p}_1 \wedge \mathbf{B}_2(\neg\mathbf{p}_2 \wedge \mathbf{k}_3\mathbf{p}_3))$	Suposição inicial: $\mathbf{B}_2(\neg\mathbf{p}_2 \wedge \mathbf{B}_1(\neg\mathbf{p}_1 \wedge \mathbf{k}_3\mathbf{p}_3))$	Suposição inicial: $\mathbf{B}_3(\neg\mathbf{p}_3 \wedge \mathbf{B}_1(\neg\mathbf{p}_1 \wedge \mathbf{k}_2\mathbf{p}_2))$
(1 vez): Nova suposição $\mathbf{B}_1(\neg\mathbf{p}_1 \wedge \mathbf{k}_2(\mathbf{p}_2 \wedge \mathbf{k}_3\mathbf{p}_3))$	(1 vez): Nova suposição: $\mathbf{B}_2(\neg\mathbf{p}_2 \wedge \mathbf{k}_1(\mathbf{p}_1 \wedge \mathbf{k}_3\mathbf{p}_3))$	(1 vez): Nova suposição: $\mathbf{B}_3(\neg\mathbf{p}_3 \wedge \mathbf{k}_1(\mathbf{p}_1 \wedge \mathbf{k}_2\mathbf{p}_2))$
(2 vez): Nova suposição $\mathbf{k}_1(\mathbf{p}_1 \wedge \mathbf{k}_2(\mathbf{p}_2 \wedge \mathbf{k}_3\mathbf{p}_3))$	(2 vez): Nova suposição: $\mathbf{k}_2(\mathbf{p}_2 \wedge \mathbf{k}_1(\mathbf{p}_1 \wedge \mathbf{k}_3\mathbf{p}_3))$	(2 vez): Nova suposição: $\mathbf{k}_3(\mathbf{p}_3 \wedge \mathbf{k}_1(\mathbf{p}_1 \wedge \mathbf{k}_2\mathbf{p}_2))$
O pai pergunta (3 vez) Responde "Sim"	O pai pergunta (3 vez) Responde "Sim"	O pai pergunta (3 vez) Responde "Sim"

Como podemos observar é possível verificar empiricamente que a formula apresentada em [4] pode representar a maneira como uma criança *i* raciocina e chega a conclusão de que está com a testa suja ou não em k passos.

6. Conclusões

Este trabalha apresentou uma semântica formal para descrever os chamados, *programas baseados em conhecimento*, baseando-se principalmente em [1] e [2]. Para isso foram apresentados os conceitos básicos de ações, protocolos e contextos, vistos em [5], e também a definição de consistência necessária para descrever execuções de um protocolo num determinado contexto. Também foi formalizada a definição de programa normal (standard) para que em cima desta apresentássemos a formalização dos programas baseados em conhecimento.

Os Programas normais não conseguem descrever a relação entre ações e o conhecimento do agente. Este problema é considerado uma das principais motivações dos autores na definição de programas baseados em conhecimento. Assim, foi formalizada uma linguagem que descreve os programas normais que dá ênfase no fato que a execução do agente está baseada nos testes aplicados em seu estado local, e estendida para os

programas baseados em conhecimento inserindo testes de conhecimento que dependem de todo o sistema interpretado.

A partir das descrições dos programas como apresentado nas seções 4 e 5, nos podemos descrever protocolos que definem o comportamento do agente em um determinado estado global do sistema. E se considerarmos um programa **Pg** como um programa normal, nós também podemos considera-lo como um programa baseado em conhecimento. Isso ocorre porque podemos associar um protocolo com **Pg** de duas maneiras: seja $I = (\mathcal{R}, \pi)$ então (1) nos podemos pensar que **Pg** é um programa normal e associa-lo ao protocolo $\mathbf{Pg}^{\pi}$; (2) ou nos podemos pensar que **Pg** é um programa baseado em conhecimento e associa-lo ao protocolo $\mathbf{Pg}^{I}$. As definições apresentadas nas seções 4 e 5 garantem que estes protocolos são idênticos.

Descrever programas no formato de KBP tem a desvantagem de não poder executar diretamente, porém pode facilitar o entendimento do problema e produzir programas mais eficientes, dependendo do contexto onde o programa será executado. O exemplo típico foi mostrado na seção 1, o problema da transmissão de um bit que usando a abordagem de KBP o Receptor para de enviar o aviso de confirmação ao Emissor, enquanto que na abordagem normal o Receptor envia o aviso infinitamente. Esta propriedade surge pelo fato que os KBP's são maneiras mais abstratas de resolvermos um problema e que, possivelmente, podem produzir nenhuma, uma, ou diversas soluções para este problema.

Dessa forma, podemos dizer que os programas baseados em conhecimento são uma forma mais eficaz de escrever programas, porém com alguns problemas como a dificuldade de passar de um KBP para uma implementação executável e o problema de representação única que não foi discutida neste no texto, mas é apresentada de forma detalhada em [1] e [2].

O propósito deste trabalho foi apresentar a intuição, formalização e dar um exemplo da modelagem de um problema utilizando os KBP's. Espero assim, ter contribuído para que esta abordagem seja compreendida e absorvida da maneira mais "amigável" possível pelo leitor e que este se sinta motivado a utiliza-la, ou mesmo, critica-la, pois só dessa forma será possível a produção de modelos mais adequados para um determinado contexto.

References

[1] Halpern, Fagin, Moses e Vardi (1995) "Knowledge-Based Programs", Symposium on Principles of Distributed Computing, 1995.

[2] Halpern, Fagin, Moses e Vardi (1995) "Reasoning about Knowledge", MIT Press, pg. 233-269.

[3] Vardi (1996) "Implementing Knowledge-Based Programs", Theoretical Aspects of Rationality and Knowledge (TARK 1996).

[4] Carlini, M. (2002) "Especificações de Sistemas Multi-Agentes Baseados em Conhecimento", Dissertação de Mestrado (UFRJ).

[5] Notas de Aula MAC5729.