UNIVERSIDADE ESTADUAL DE PONTA GROSSA
SETOR DE CIÊNCIAS AGRÁRIAS E DA TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

ALCEU DE SOUZA BRITTO DOS SANTOS
MATEUS FELIPE DA SILVA JUNGES
RAFAEL DE MATTOS
VICTOR PIOTROVSKI BEGHA

EXERCÍCIOS INFERÊNCIA PROPOSICIONAL

PONTA GROSSA
2019

Exercícios Inferência Proposicional

1) Consulte um dos seguintes livros:

Inteligência Artificial de Stuart Russell e Peter Norvig (qualquer uma das edições)

Bratko, I. Artificial Intelligence

Leia a seção sobre lógica proposicional (sistemas baseados em regras/cláusulas e Horn).

2) O que é uma cláusula de Horn/Sistema baseado em regras?

Uma resolução completa de um problema torna um método de inferência muito importante. Em muitas situações práticas, entretanto, não é necessário ter todo poder da solução do problema. Algum conhecimento básico do mundo real satisfaz certas restrições contidas em forma de sentença, das quais possibilitam fazer o uso de algoritmos de inferência mais restritivos e eficientes.

A cláusula Horn é uma disjunção de literais dos quais um é positivo. Então todas as cláusulas definidas são cláusulas de Horn, assim como cláusulas que não possuem literais positivos; essas são chamadas de cláusulas de meta/objetivo. Cláusulas de Horn são fechadas para a seguinte resolução: se você resolve duas cláusulas de Horn, você retorna uma cláusula de Horn.

$$\neg P_1 \vee \neg P_2 \vee \cdots \neg P_k \vee Q$$

Que é equivalente a:

$$(P_1 \wedge P_2 \wedge \cdots P_k) \rightarrow Q$$

3) Descreva o encadeamento para frente. Qual a conexão com grafos e-ou?

Um algoritmo de encadeamento para frente determina se um único símbolo proposicional *q* - a busca - é implicada por um conhecimento básico de definição de cláusulas. Isso parte do conhecimento de fatos conhecidos (literais positivos) em um conhecimento básico. Se todas as premissas de uma implicação são conhecidas,

então a conclusão é adicionada aos fatos conhecidos. Em grafos E-OU, vários links são unidos por um arco que indica uma conjunção - cada link deve ser provado - enquanto múltiplos links sem arco indicam uma disjunção - qualquer link deve ser provado.

4) Descreva o encadeamento para trás. Qual a conexão com uma árvore e-ou?

O algoritmo de encadeamento para trás, como o próprio nome já sugere, funciona atrás da busca. Se a busca *q* é conhecida por ser verdadeira, então nenhum trabalho será necessário. Por outro lado, o algoritmo encontra aquela implicações no conhecimento básico das quais resultam em *q*. Se todas as premissas de uma das implicações for provada como verdade (por encadeamento para trás), então *q* é verdadeiro.

5) Estes algoritmos são consistentes e completos?

Os algoritmos de encadeamento para frente ou para trás podem ser considerados completos em razão de que, nestes algoritmos, toda sentença atômica válida pode ser derivada usando o sistema, a partir do conhecimento existente. Além disso, os algoritmos são considerados consistentes por não resultarem em contradições, usando sempre Modus Ponens em sequência.

6) Qual a função das variáveis agenda e contagem (counting) no algoritmo de encadeamento para frente descrito no texto de Russell e Norvig?

A variável “agenda” é uma lista de símbolos, a qual é inicializada como sendo a lista dos símbolos verdadeiros dentro da base de conhecimento; os elementos dessa lista serão verificados um a um, e em seguida removidos, até que “agenda” esteja vazia; também são adicionados à lista, ao longo do processo, as novas conclusões que são reconhecidas como verdadeiras.

A variável “contagem” é uma tabela que armazena, para cada cláusula “c” existente, o número de símbolos nas premissas desta cláusula - isso é usado para verificar se a premissa sendo analisada na atual iteração possui uma conclusão associada a ela.

7) Por que diz-se que o encadeamento para frente é orientado a dados e o encadeamento para trás é orientado por objetivos?

O encadeamento para frente é dito orientado a dados devido ao fato que o raciocínio em que é baseado parte dos dados conhecidos, isto é, ele é usado para derivar conclusões a partir das informações de entrada, sem necessariamente ter uma conclusão específica em mente; em contrapartida, o encadeamento para trás é orientado por objetivos porque a conclusão desejada é conhecida desde o princípio, a partir da qual o algoritmo buscará implicações (dentro da sua base de conhecimento atual) que levarão até esta conclusão.

8) Defina o problema da satisfazibilidade (satisfatibilidade)? Qual a conexão deste problema com o conceito de prova por refutação e consequência lógica? O que é a transição de fase/ponto crítico neste problema?

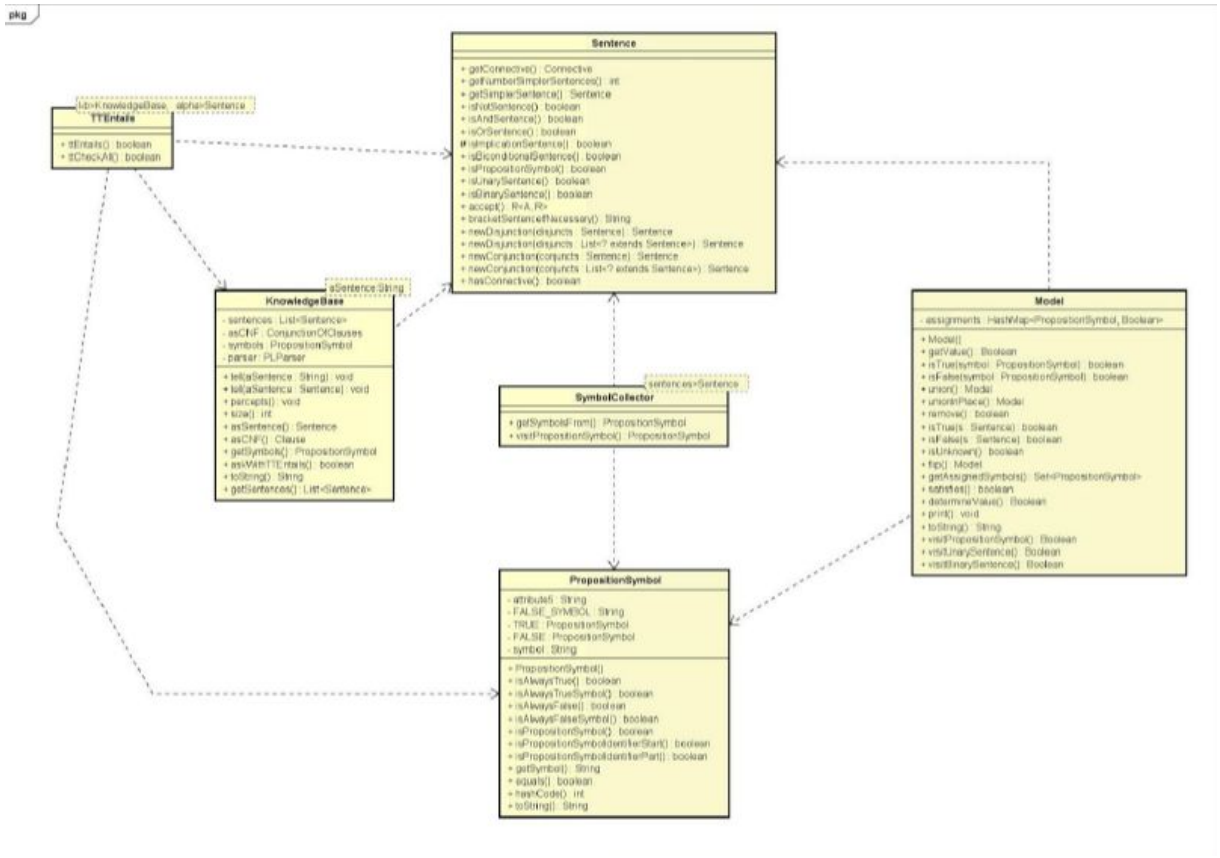
Uma sentença é dita satisfatível se existir algum modelo que a torne verdadeira; dessa forma, o problema de satisfatibilidade se refere ao problema de determinar se sentenças na lógica proposicional são satisfatíveis, ao verificar se existe pelo menos uma condição que torne verdadeiro o problema ou sentença em questão. Esse conceito se conecta com a prova por refutação e consequência lógica porque a prova por refutação, também chamada prova por contradição, supõe que uma determinada sentença A apenas implicará tautologicamente em uma outra sentença B se a proposição (A ^ ~B) for não satisfazível, usando então a satisfatibilidade para verificar esta condição. Já a transição de fase ou ponto crítico deste problema se refere à afirmação que, se existirem um número “n” de variáveis, à medida que este número tende ao infinito, existe um ponto onde instâncias

superiores serão satisfatíveis e instâncias inferiores serão insatisfatíveis - ou seja, existe um limiar de satisfatibilidade onde ocorre uma transição entre problemas satisfatíveis e insatisfatíveis.

9) Execute e descreva os testes e os métodos implementados na classe TTEntailsTest.

A classe TTEntailsTest executa o algoritmo de lógica proposicional. Inicialmente ele testa duas sentenças simples A e B da base de conhecimento utilizando o procedimento testSimpleSentence1() da classe TTEntailsTest e passando como parâmetro “A & B” através do procedimento tell() da KnowledgeBase, depois passa como parâmetro para através do procedimento assertEquals() da classe Assert os valores true e o simbolo “A” passado por inferência através do procedimento askWithTTEntails() da classe KnowledgeBase. Posteriormente o programa testa duas sentenças simples A e B da base de conhecimento utilizando o procedimento testSimpleSentence2() da classe TTEntailsTest e passando como parâmetro “A | B” através do procedimento tell() da KnowledgeBase, depois passa como parâmetro para através do procedimento assertEquals() da classe Assert os valores true e o simbolo “A” passado por inferência através do procedimento askWithTTEntails() da classe KnowledgeBase. Em seguida o programa testa duas sentenças simples A e B da base de conhecimento utilizando o procedimento testSimpleSentence3() da classe TTEntailsTest e passando como parâmetro “(A => B) & A” através do procedimento tell() da KnowledgeBase, depois passa como parâmetro para através do procedimento assertEquals() da classe Assert os valores true e o simbolo “A” passado por inferência através do procedimento askWithTTEntails() da classe KnowledgeBase.

10) Elabore um diagrama de classes relativo às classes TTEntails, PropositionSymbol, SymbolCollector, KnowledgeBase, Model e Sentence. Descreva o que estas abstrações representam no problema da inferência lógica.

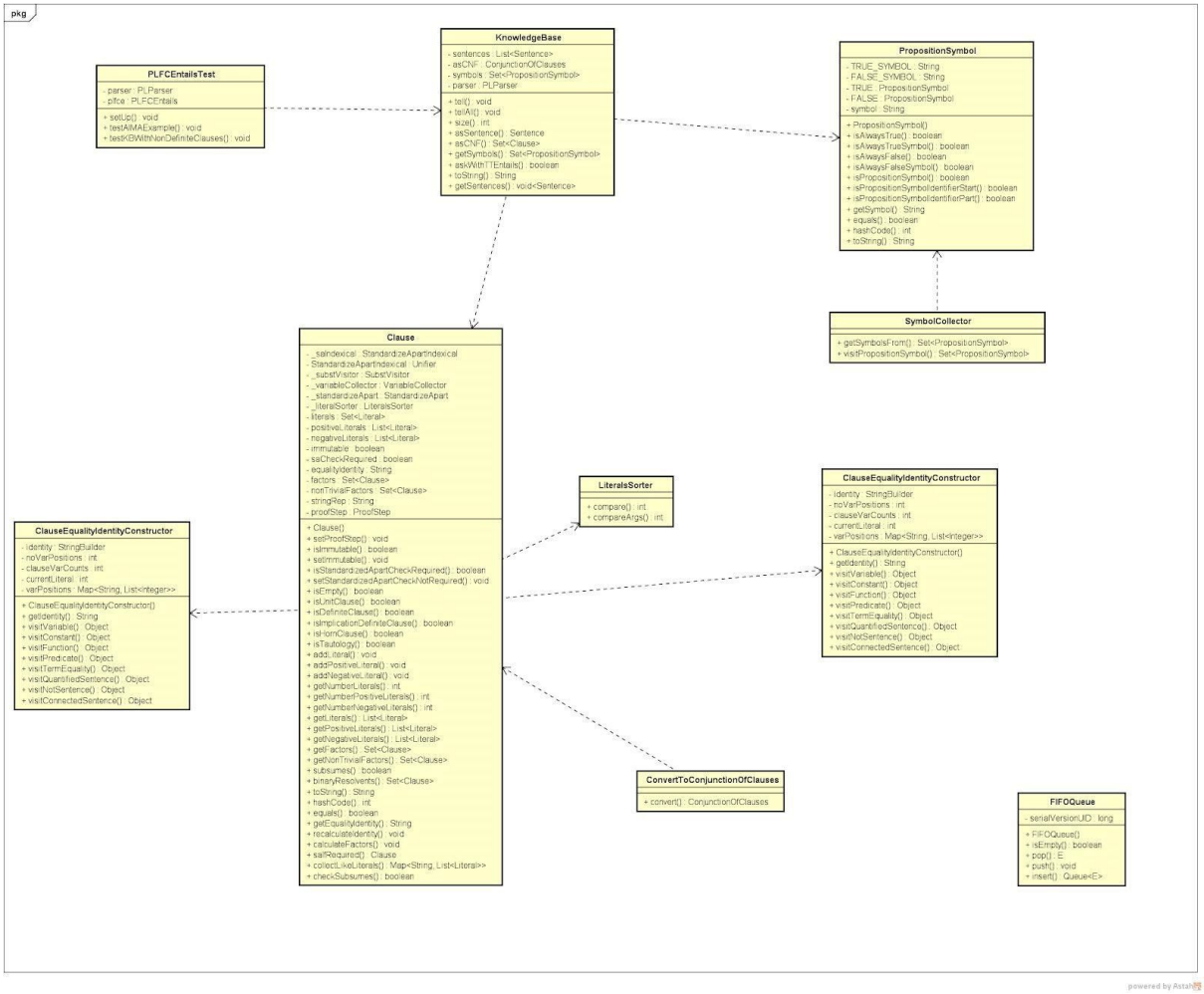


11. Execute e descreva os testes e os métodos implementados na classe PLFCEntailsTest.

A classe PLFCEntailsTest pertence ao pacote de algoritmos de lógica proposicional por análise e por inferência, e possui os métodos setUp() que inicia as variáveis parser e plfce e o método testAIMAExample() que testa as cláusulas da Base de Conhecimento KnowledgeBase, cria o simbolo proposicional q e passa como parâmetro através do método assertEquals() da classe Assert os valores true e a variável kb e o símbolo q, e o método testKBWithNonDefiniteClauses() verifica se existe alguma cláusula que não está definida na KnowledgeBase , cria o simbolo

proposicional q e passa como parâmetro através do método assertEquals() da classe Assert os valores true e a variável kb e o símbolo q.

12) Elabore um diagrama de classes relativo às classes PLFCEntails, KnowledgeBase, Clause, PropositionSymbol, ConvertToConjunctionOfClauses, SymbolCollector, FIFOQueue do pacote aima.core.logic.propositional. Descreva o que estas abstrações representam no problema da inferência lógica.



13.

A	B	¬B	A->B	A ^ ¬B	A |= B
V	V	F	V	F	V
V	F	V	F	V	*
F	V	F	V	F	V
F	F	V	V	F	V

a) **I -** Se A |= B então A->B = Tautologia

Se A|=B então sempre B é verdadeiro quando A é verdadeiro. Neste caso A->B = V. Quando A é falso, A->B = V.

Se o A implica logicamente em B, significa que a segunda linha da tabela nunca ocorre. Logo, a expressão é uma tautologia, visto que para cada valoração possível a resposta é verdadeira.

II - Se A->B = Tautologia então A |= B

Uma vez que A -> B = V, então todas as linhas da tabela verdade de A -> B também são verdadeiras. Logo, não ocorre A = V e B = F, mas sempre que A é igual a V, teremos B = V, que é a definição de A |=B

*: Este espaço não contém valor lógico pois seria FALSO, e na implicação lógica esse valor não existe.

b) **I -** Se A |= B então A^¬B = Contradição

Se A|=B então sempre B é verdadeiro quando A é verdadeiro. Neste caso A^¬B = F. Quando A é falso, A^¬B = F.

Se o A implica logicamente em B, significa que a segunda linha da tabela nunca ocorre. Logo, a expressão é uma contradição, visto que para cada valoração possível a resposta é falso.

II - Se A^¬B = Contradição então A |= B

Uma vez que A^¬B = F, então todas as linhas da tabela verdade de A^¬B também são falsas. Logo, não ocorre A = V e B = F, mas sempre que A é igual a V, teremos B = V, que é a definição de A |=B