JTMS - modelizado em Owl 2.0 e SWRL

Engenharia do Conhecimento

Grupo 018
João Rodrigues, nº 45582
Simão Neves, nº 45681

Índice

Introdução

O objectivo deste trabalho foi modelizar um sistema JTMS recorrendo ao Protegé para manipulação de uma ontologia em OWL e regras SWRL, podendo assim inferir através do motor de inferência, o que acreditamos ser verdade com base em conhecimento que é inserido no sistema e no modelo desenvolvido.
De modo a poder estabelecer relações binárias lógicas entre individuals de acordo com a definição de JTMS decidimos definir Classes base e suas classes de equivalência. As classes presentes neste projecto são a Node, NodeIN, NodeDedução, NodePremissa, NodeSuposicao, NodeOUT, Justification, JustificationVALID, JustificationINVALID, List, ListEMPTY, ListIN, ListIN_VALID, ListIN_INVALID, ListOUT, ListOUT_VALID, ListOUT_INVALID. De uma forma mais detalhada podemos dizer que as classes Node, Justification e List são disjuntas entre si, que as classes *VALID e *INVALID são igualmente disjuntas entre si e que uma ListEMPTY é sempre uma list VALID. As object properties (relações) definidas são hasAntecedente, hasAntepassado, hasConsequencia, hasConsequenciaAfectada, hasConsequenciaAcreditada, hasFundamento, hasJustification, hasList, hasListIN, hasListOUT, hasNode, hasRepercussao, hasRepercussaoAcreditada, hasSupportNode, isSupportedBy. Algumas propriedades inversas foram igualmente definidas para que algumas das queries efectuadas pudessem ser mais fácilmente legíveis e intuitivas. Para uma manipulação mais “manual” foi utilizada a syntax de manchester.
A nossa ontologia conta com 311 axiomas, 18 classes, 19 object properties e 42 individuals.
São em seguida apresentadas as regras SWRL utilizadas e alguns exemplos de queries passiveis de serem realizadas e respectivos resultados.

Regras SWRL utilizadas

Regra 1: **Node(?x) ^ isSupportedBy(?x, ?ju) ^ hasList(?ju, ?li) ^ hasNode(?li, ?ant) -> hasAntecedente(?x, ?ant)**
Descrição: Se um nó X, que tem uma justificação de suporte JU que tenha o nó ANT numa das suas listas, então esse nó ANT é antecedente do nó X.

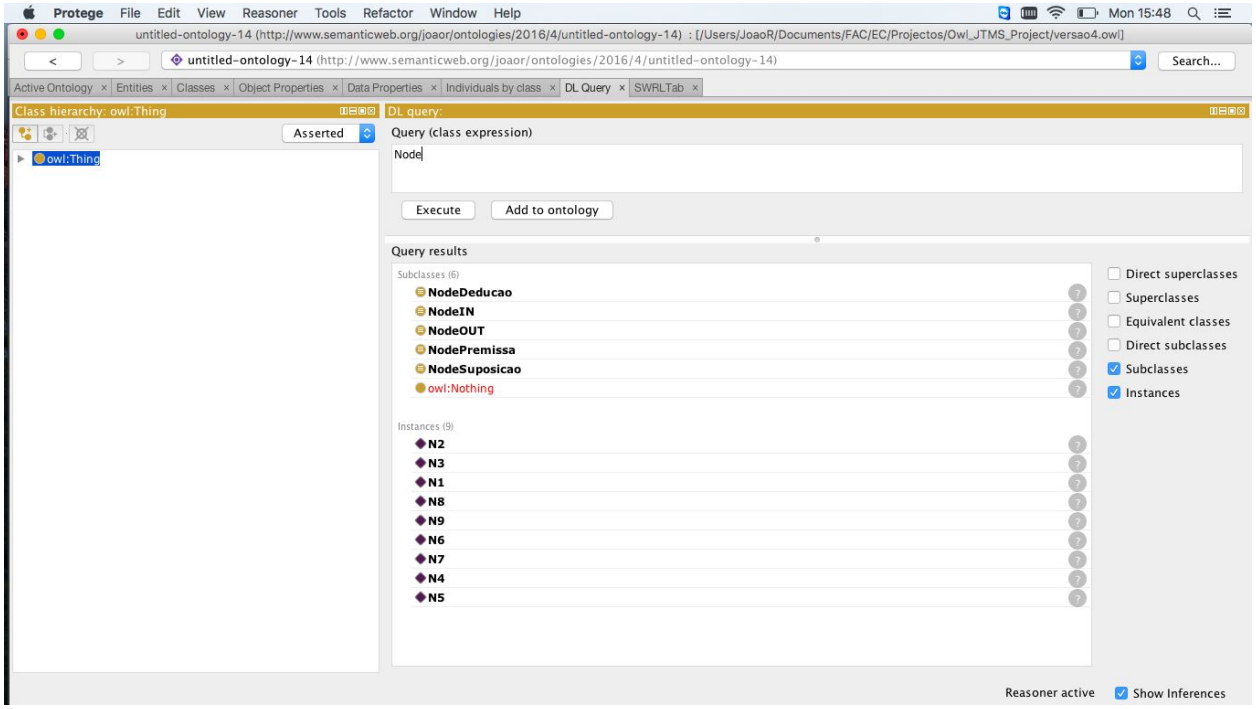
Regra 2: **Node(?x1) ^ Node(?x2) ^ differentFrom(?x1, ?x2) ^ hasJustification(?x2, ?ju) ^ hasList(?ju, ?l) ^ hasNode(?l, ?x1) -> hasConsequencia(?x1, ?x2)**
Descrição: Se houver um nó X1 diferente do nó X2, e X2 tiver uma justificação que tem X1 numa das suas listas, então X1 tem como consequência o nó X2.

Regra 3: **Node(?x) ^ Node(?y) ^ hasConsequenciaAfectada(?x, ?y) ^ NodeIN(?y) -> hasConsequenciaAcreditada(?x, ?y)**
Descrição: Se houver um nó X diferente do nó com estado IN Y, e X tem como consequência afectada Y, então o nó X tem como consequência acreditada Y.

Perguntas e respostas de queries

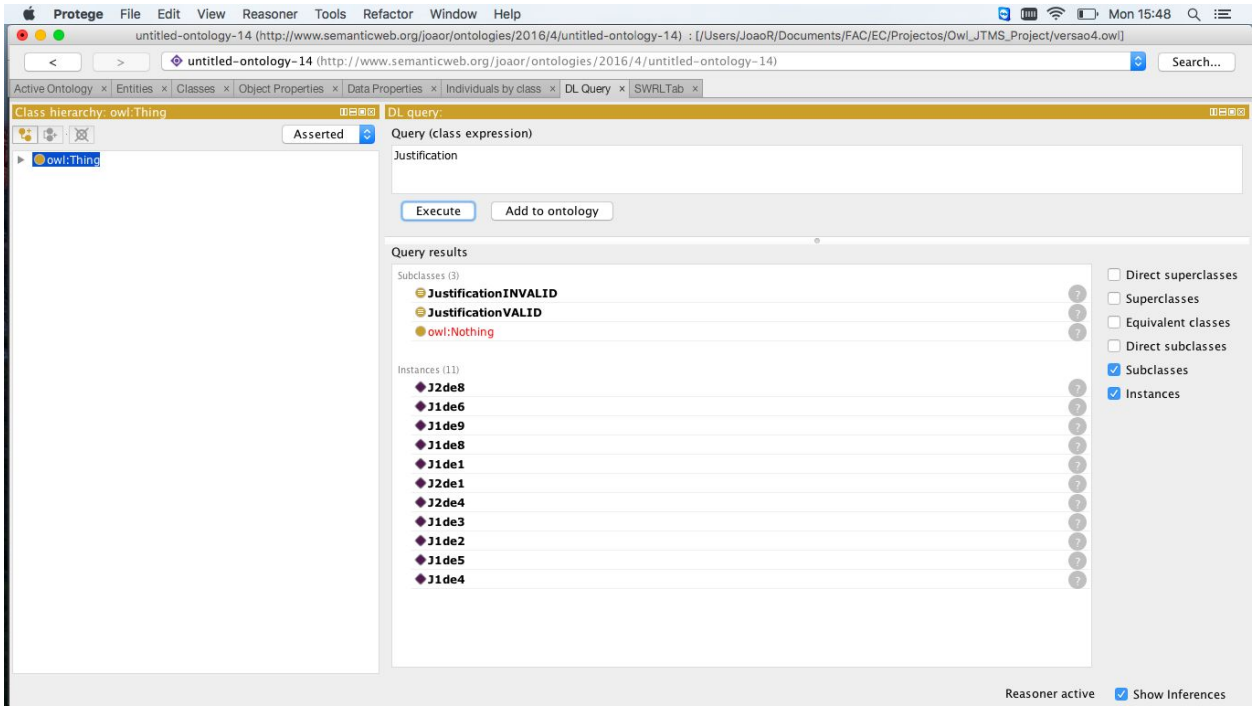
Pergunta 1: Quais os Nós?

Query: Node



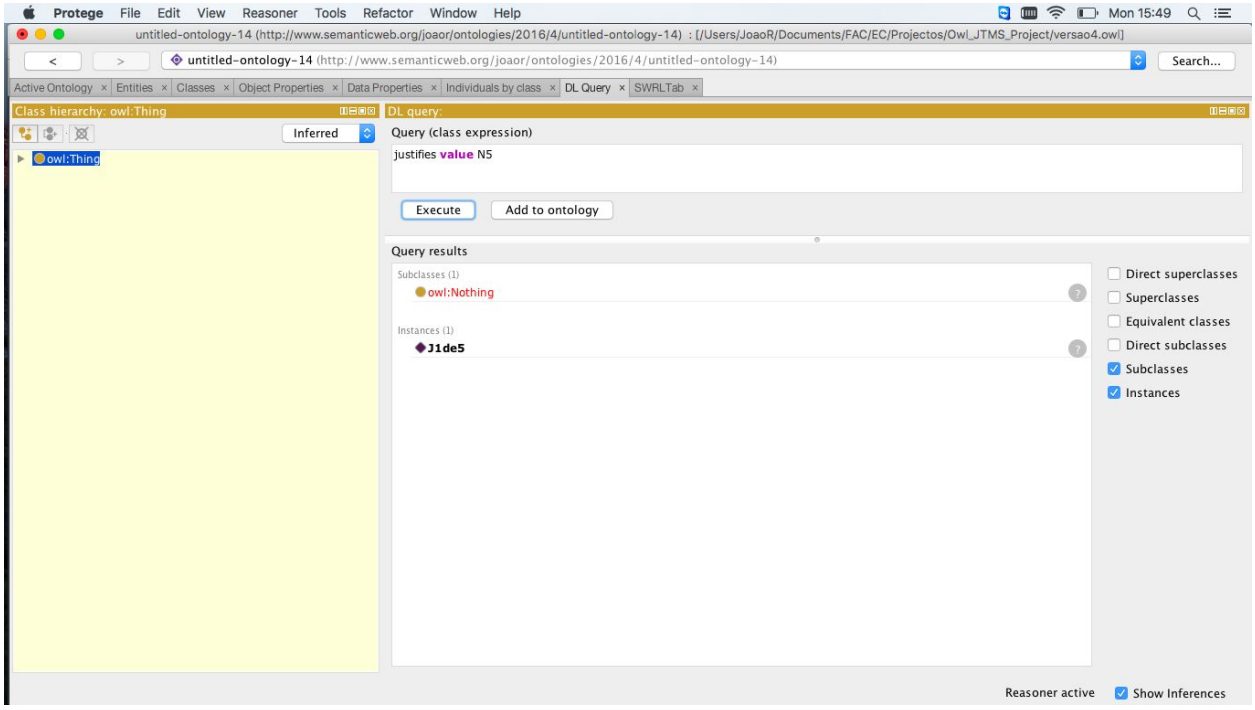
Pergunta 2: Quais as Justificações?

Query: Justification



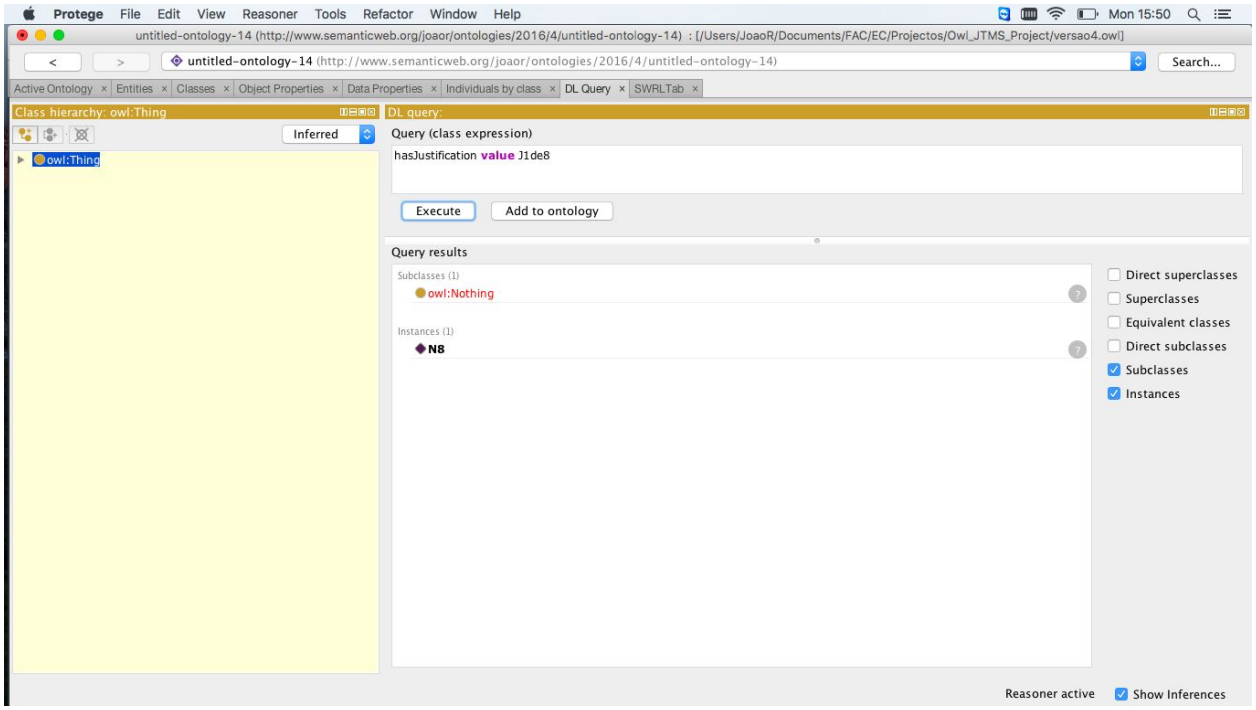
Pergunta 3: Quais as justificações de N5?

Query: Justifies value N5



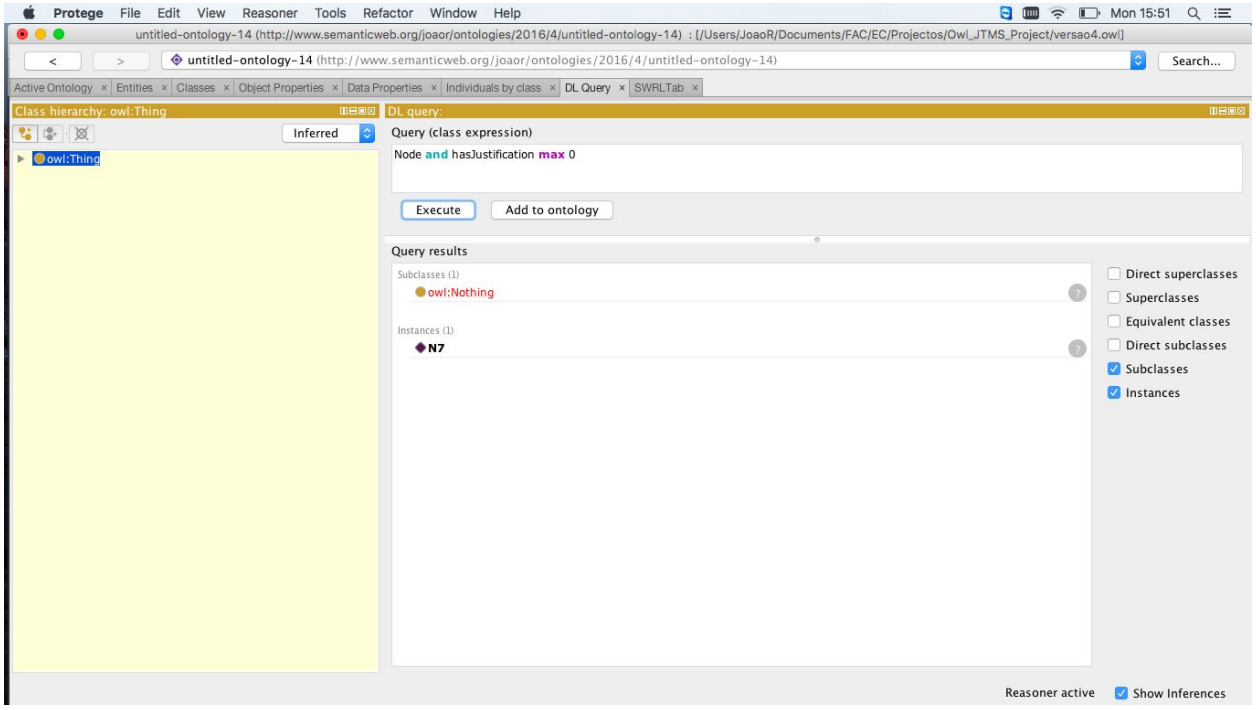
Pergunta 4: Qual o Nó que possui a justificação de J1de8?

Query: hasJustification value J1de8



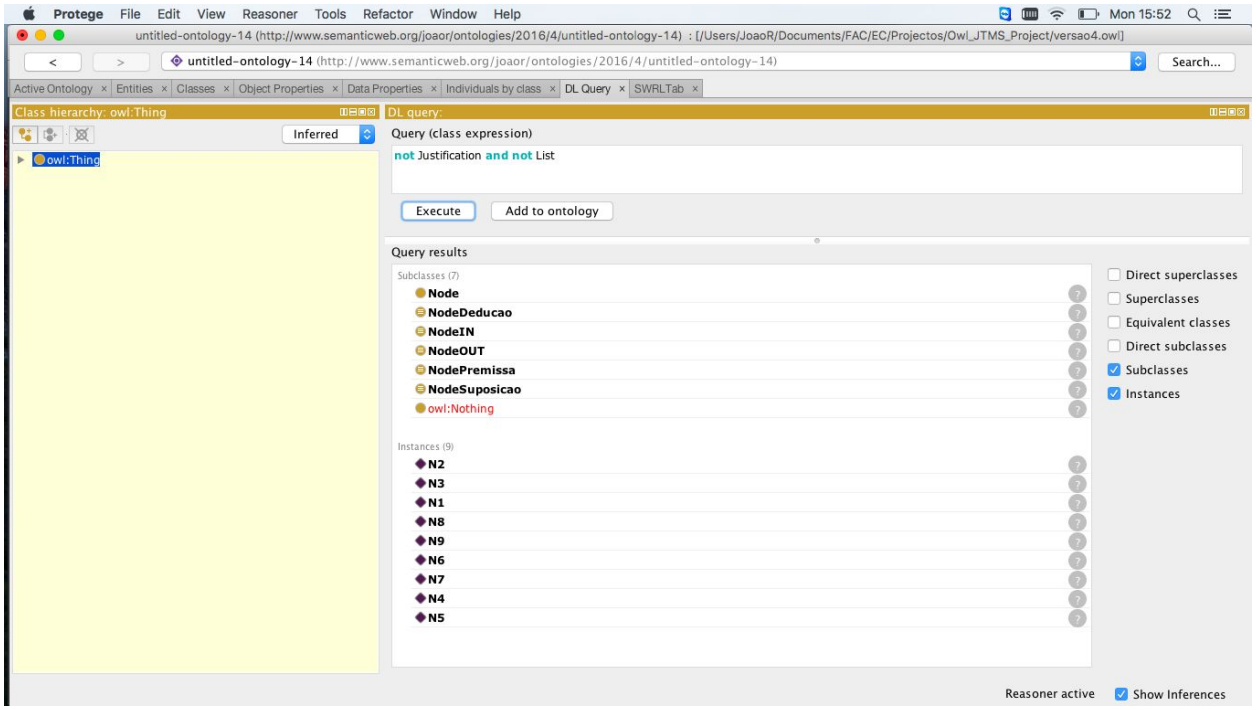
Pergunta 5: Quais os Nós sem justificação?

Query: Node and hasJustification max 0



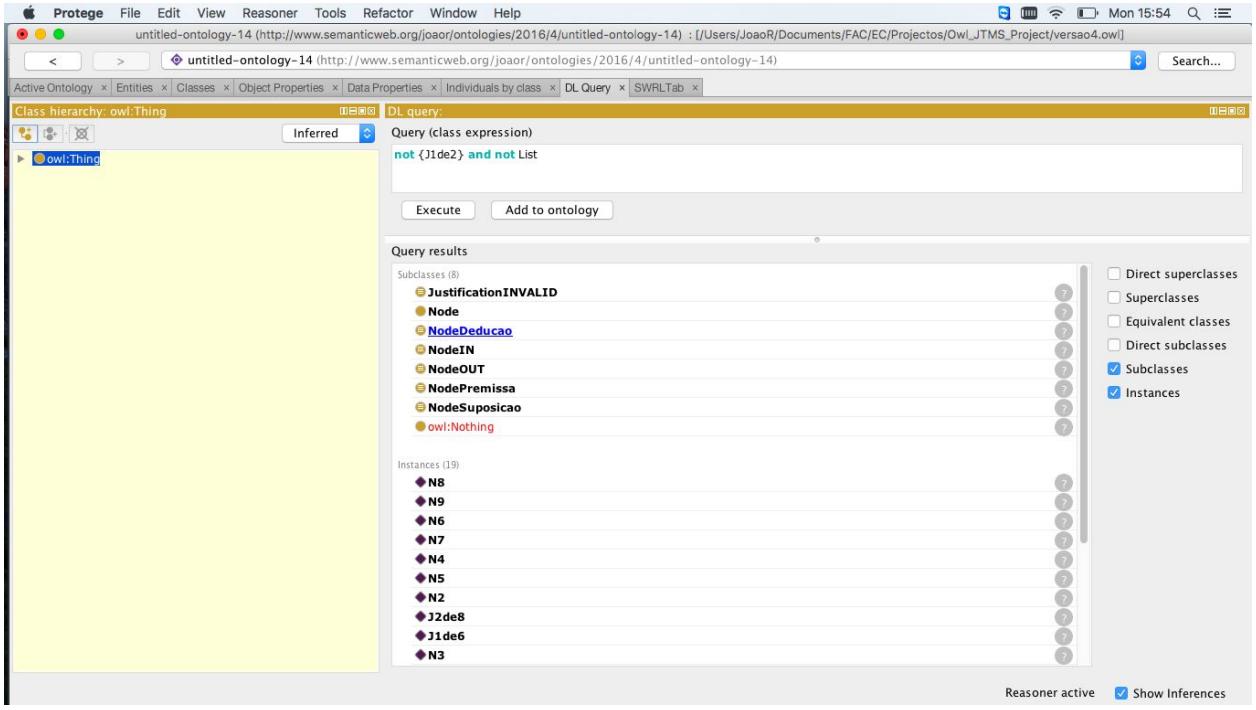
Pergunta 6: Quem não é uma justificação?

Query: Not Justification and not List



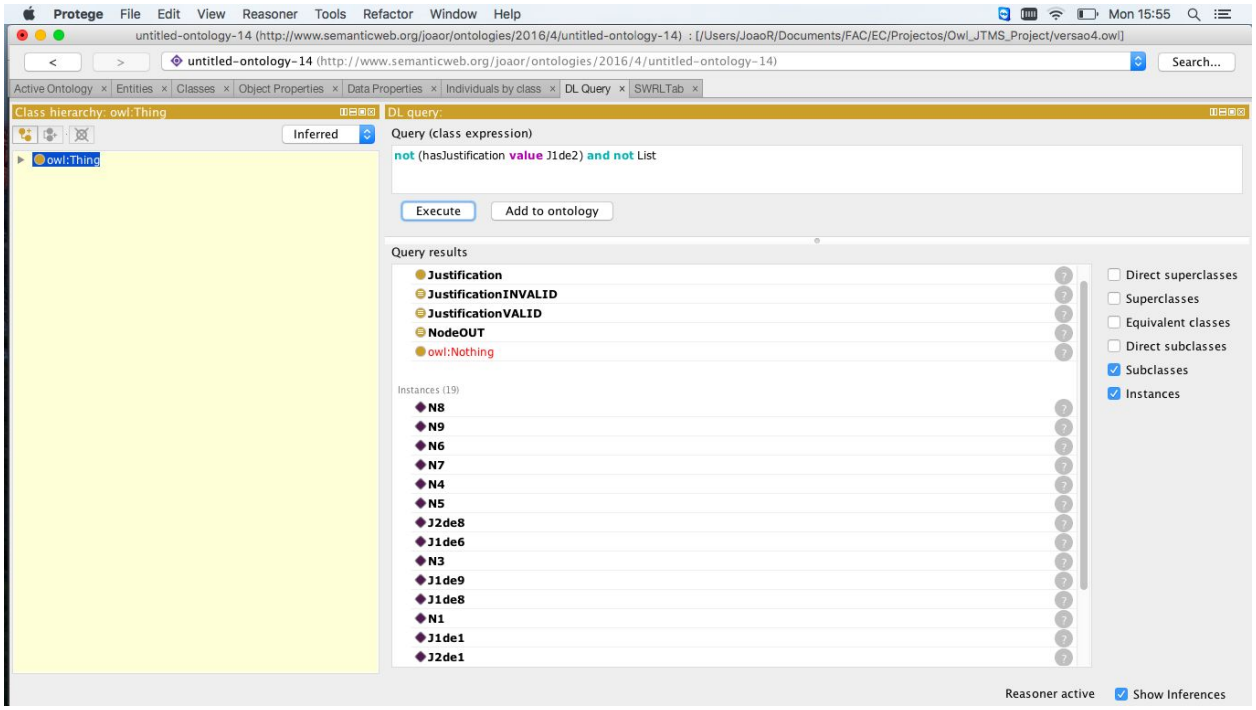
Pergunta 7: Quem não é o J1de2?

Query: Not {J1de2} and not List



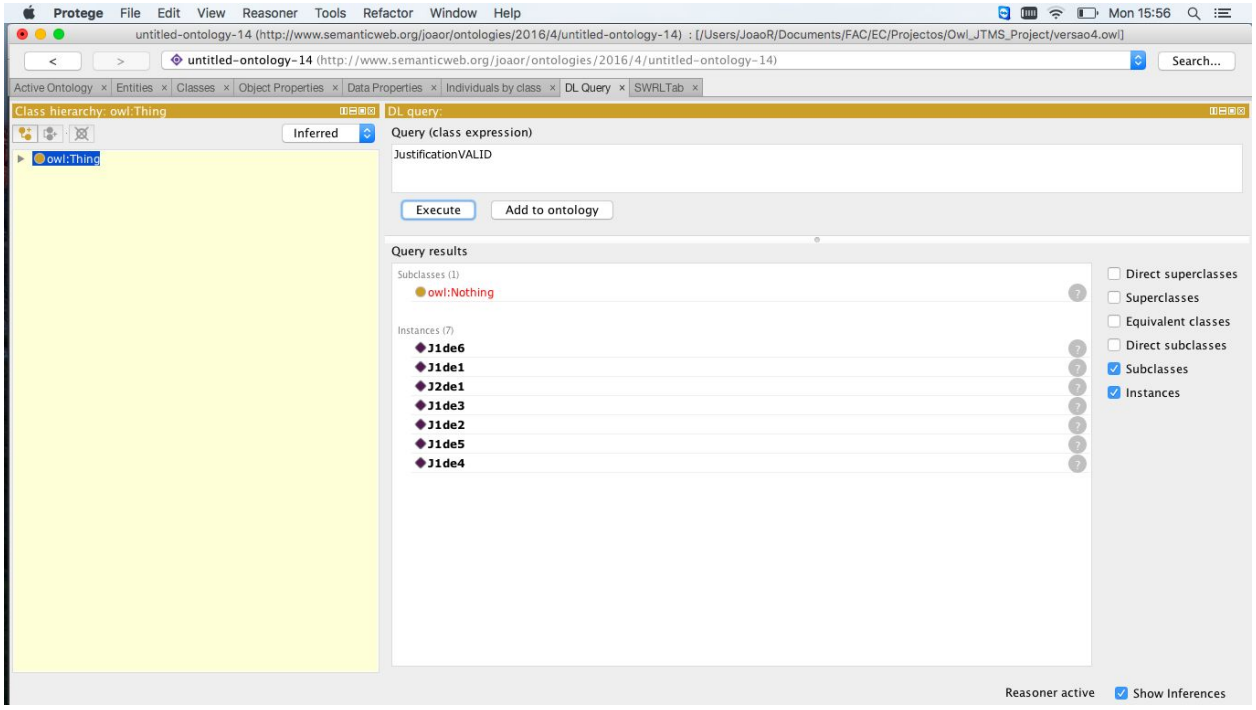
Pergunta 8: Quem não possui a justificação J1de2?

Query: Not(hasJustification value J1de2) and not List



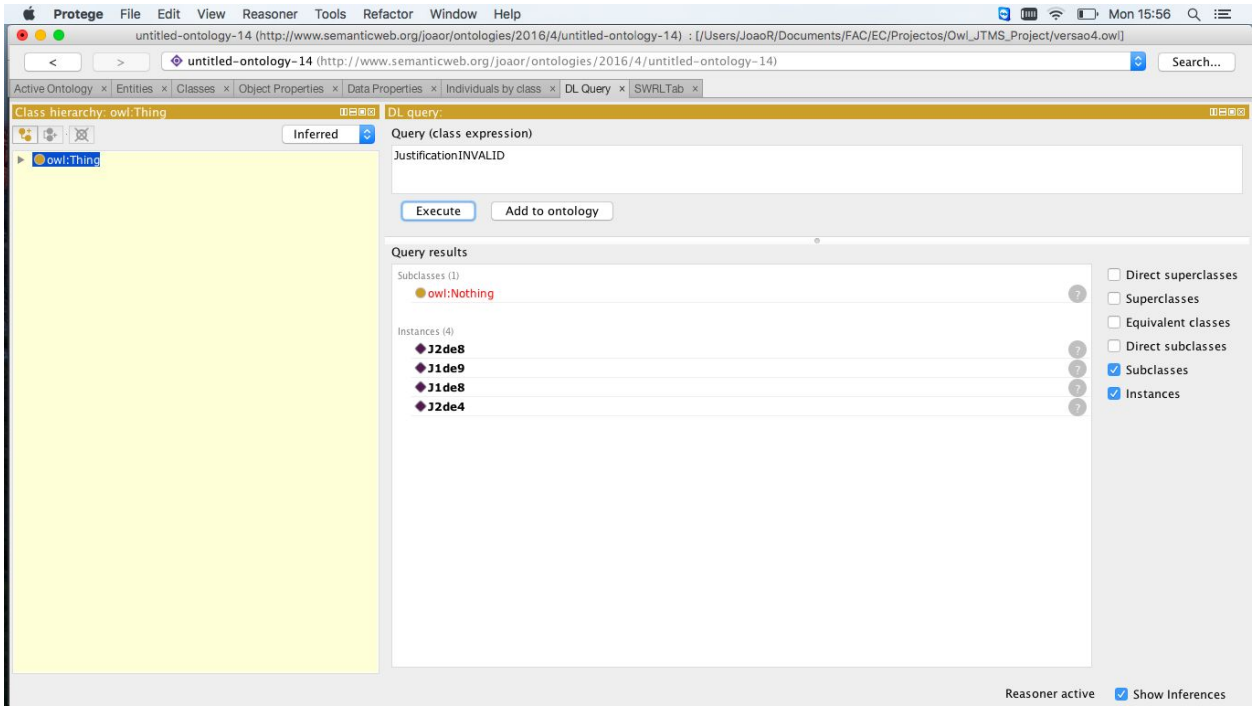
Pergunta 9: Quem é justificação válida?

Query: JustificationVALID



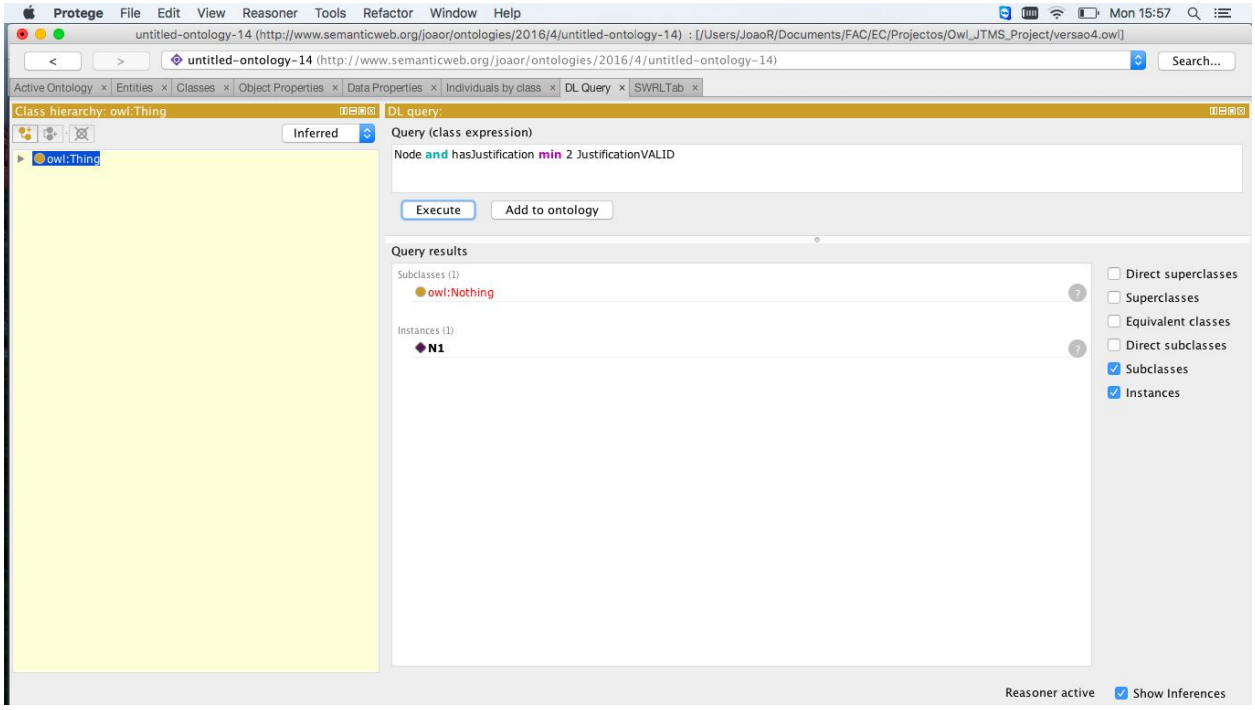
Pergunta 10: Quem é justificação inválida?

Query: JustificationINVALID



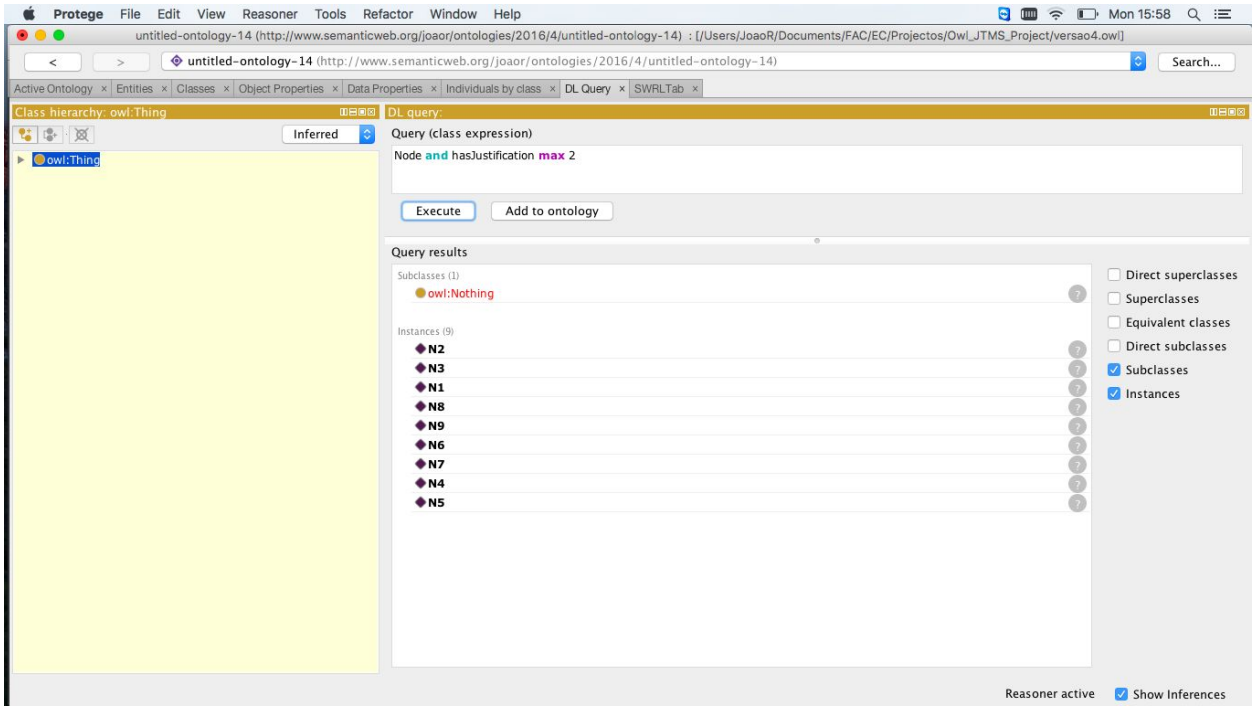
Pergunta 11: Quais os nós que têm pelo menos duas justificações válidas?

Query: Node and hasJustification min 2 JustificationVALID



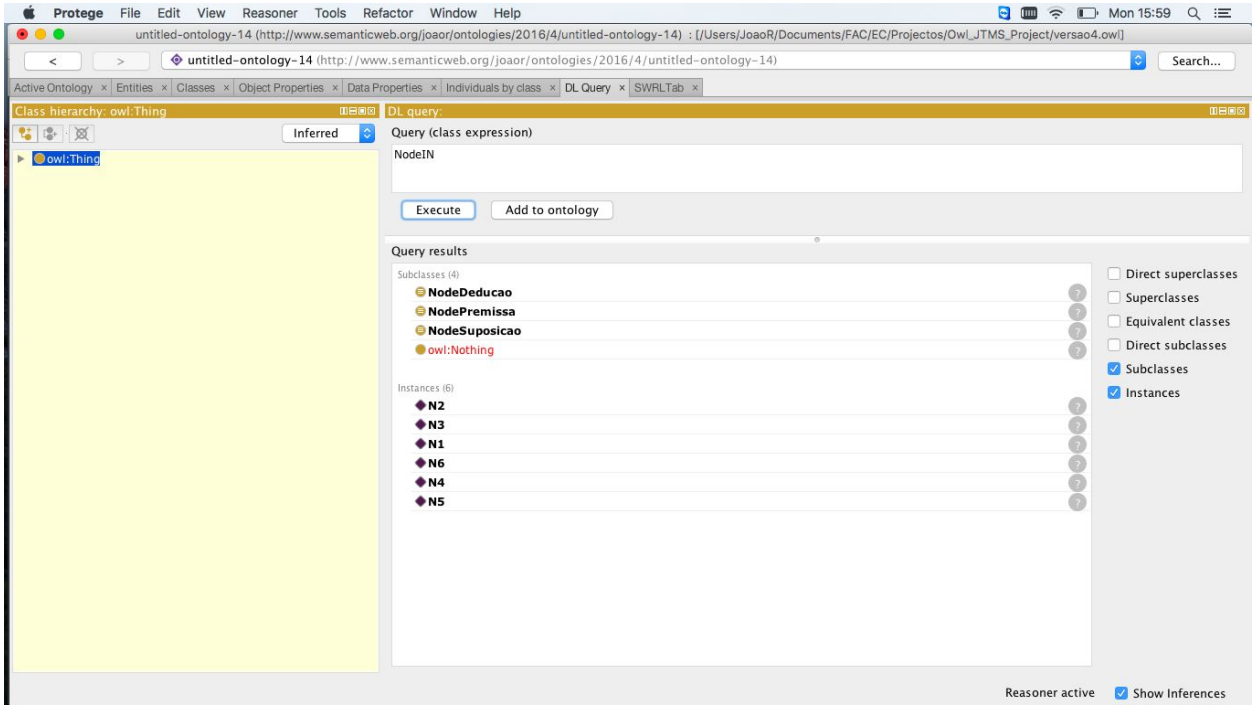
Pergunta 12: Quais os nós que não têm 3 ou mais justificações?

Query: Node and hasJustification max 2



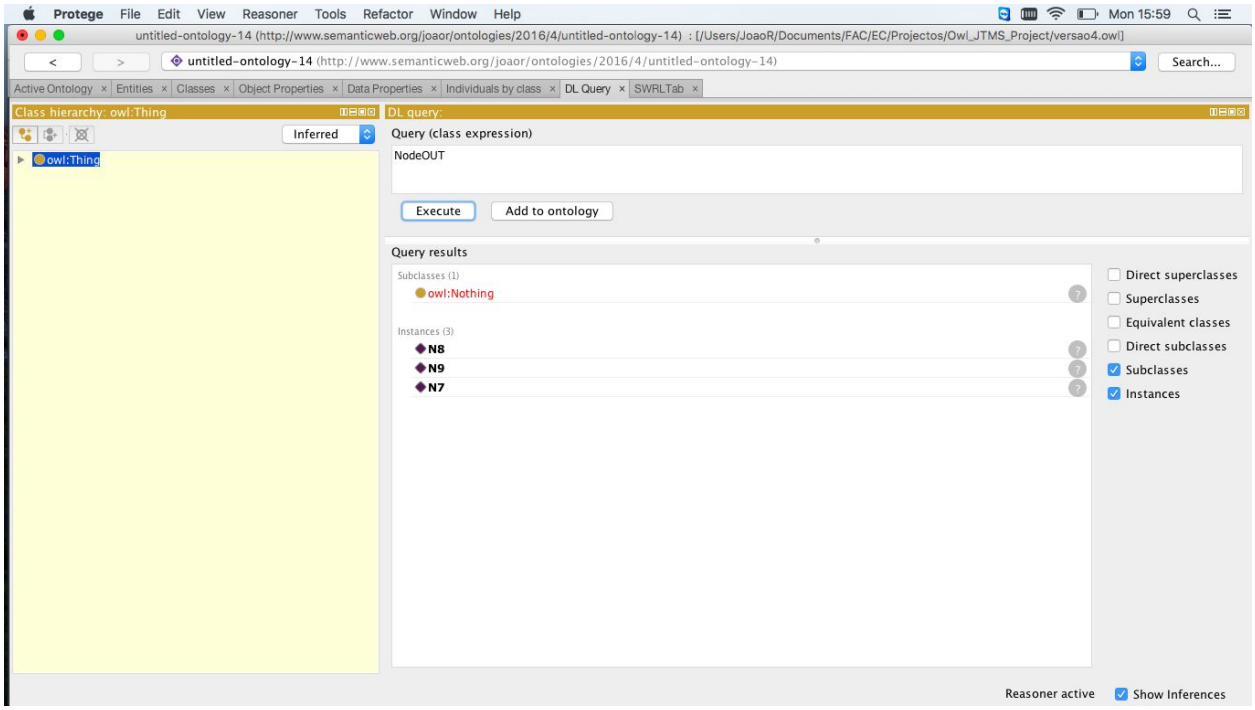
Pergunta 13: Quais os nós com estatuto IN?

Query: NodeIN



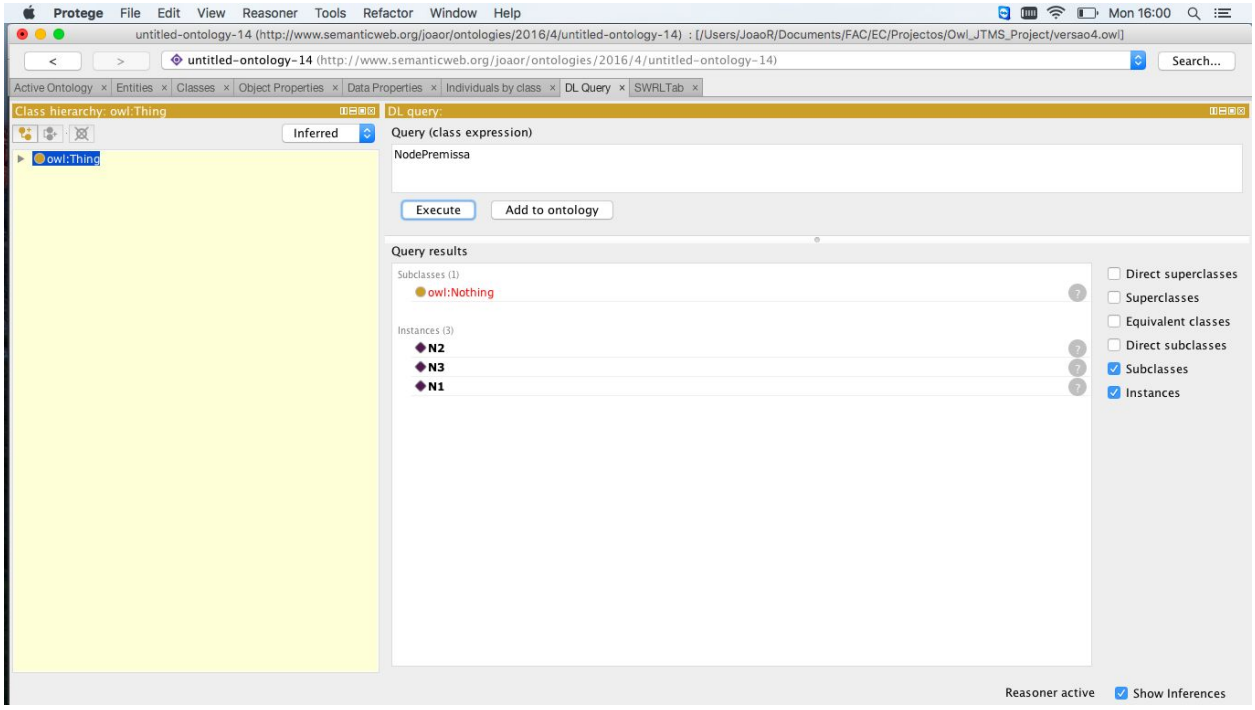
Pergunta 14: Quais os nós OUT?

Query: NodeOUT



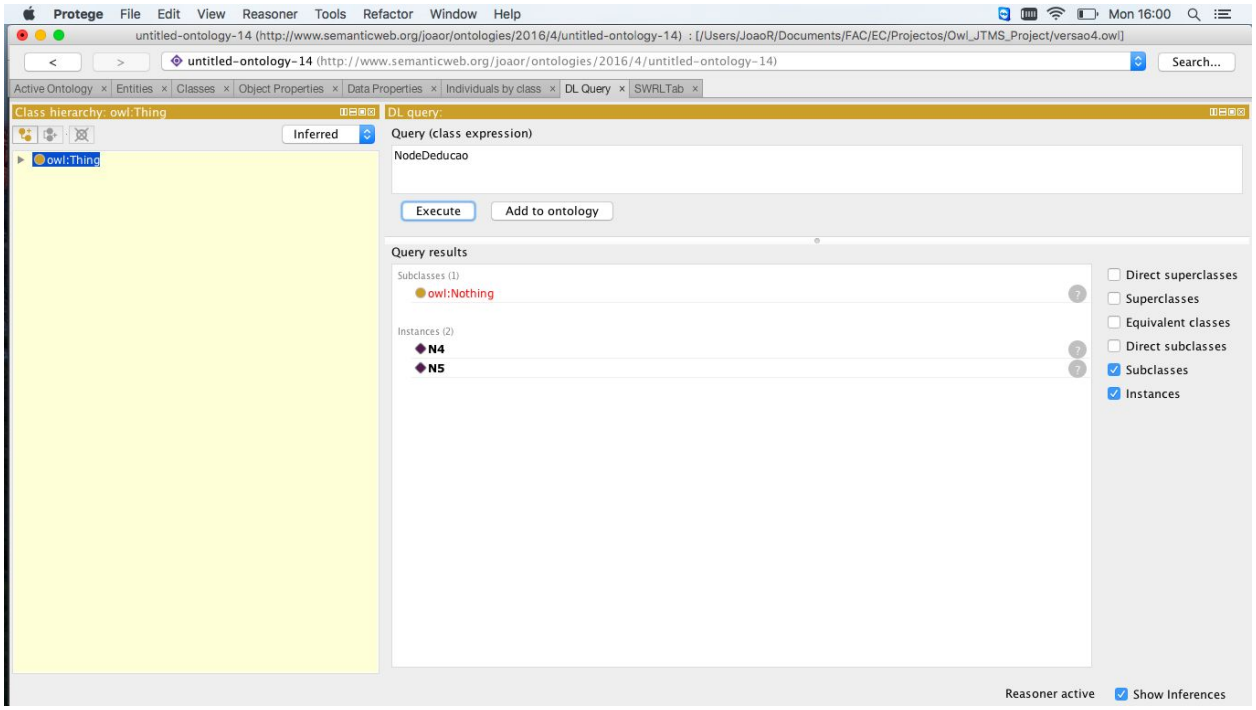
Pergunta 15: Quais as premissas?

Query: NodePremissa



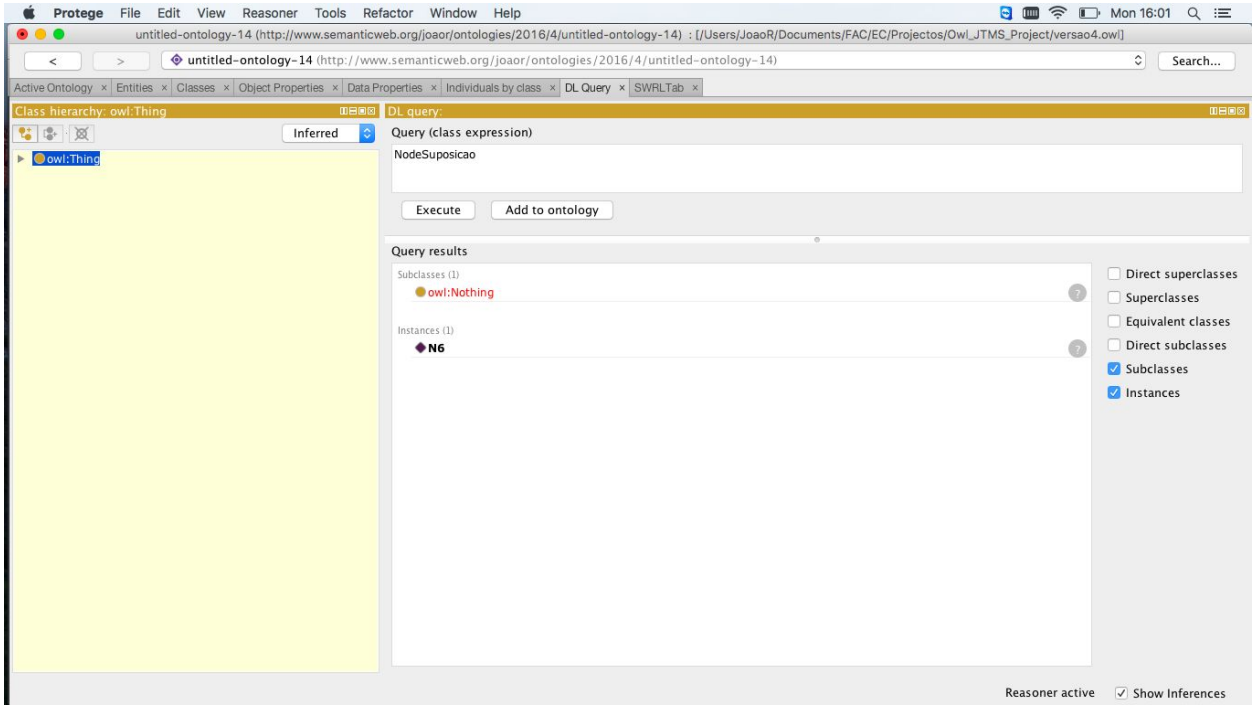
Pergunta 16: Quais as deduções?

Query: NodeDeducao



Pergunta 17: Quais as presuposições?

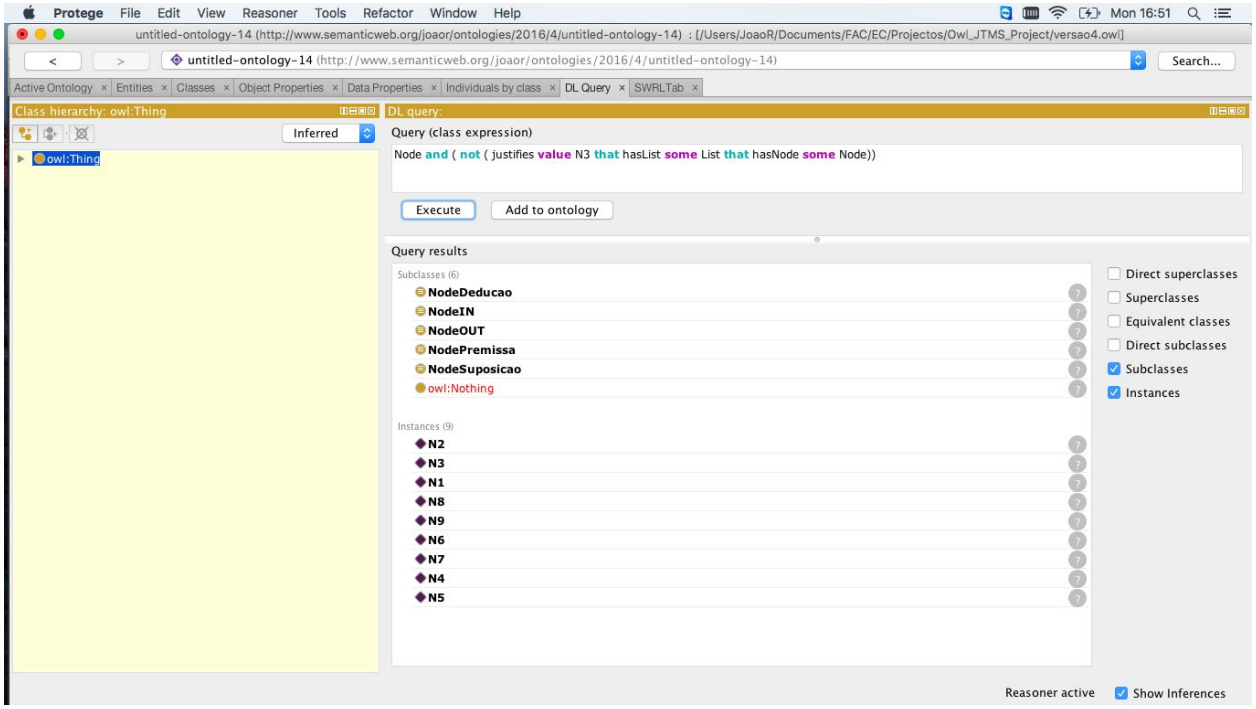
Query: NodeSuposicao



Pergunta 18: Dado o modelo apresentado foram efectuadas algumas queries para a sua validação. Para isso definimos propriedades inversas apenas por uma questão de lógica de query. Todas as queries efectuadas devolveram os valores esperados.

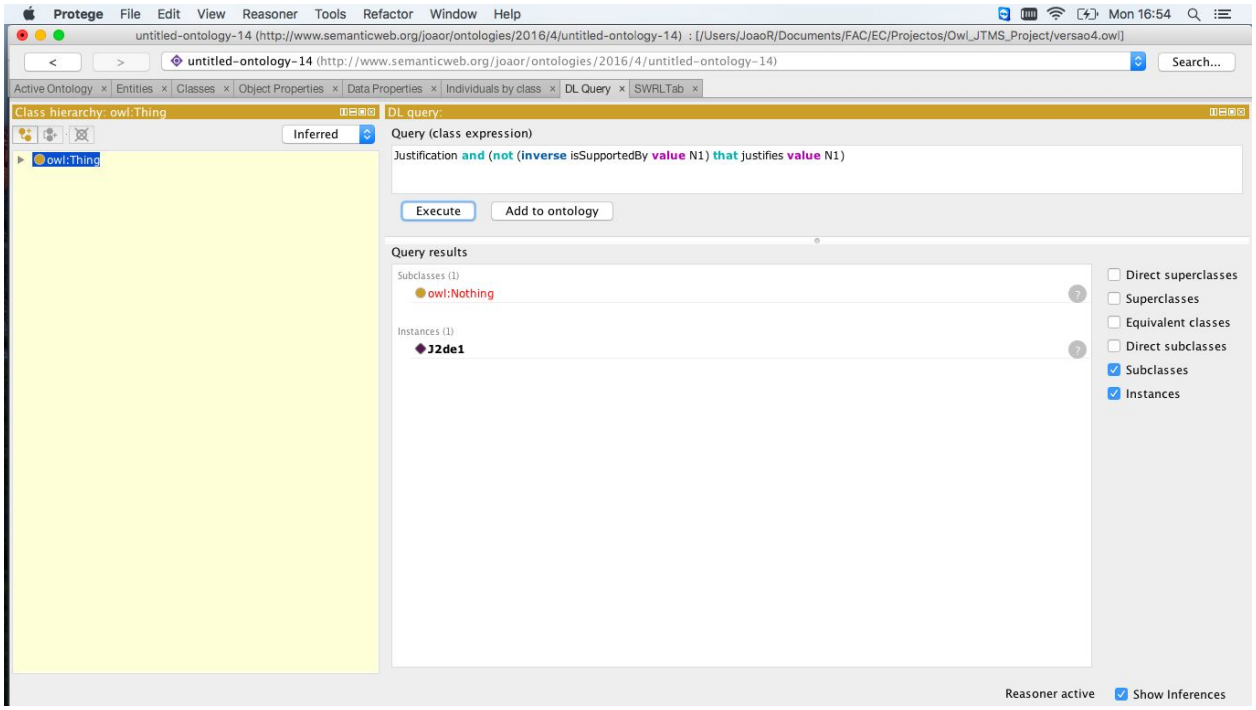
Pergunta 19: Quais os nós que não são suporte de N3?

Query: Node and (not (justifies value N3 that hasList some List that hasNode some Node))



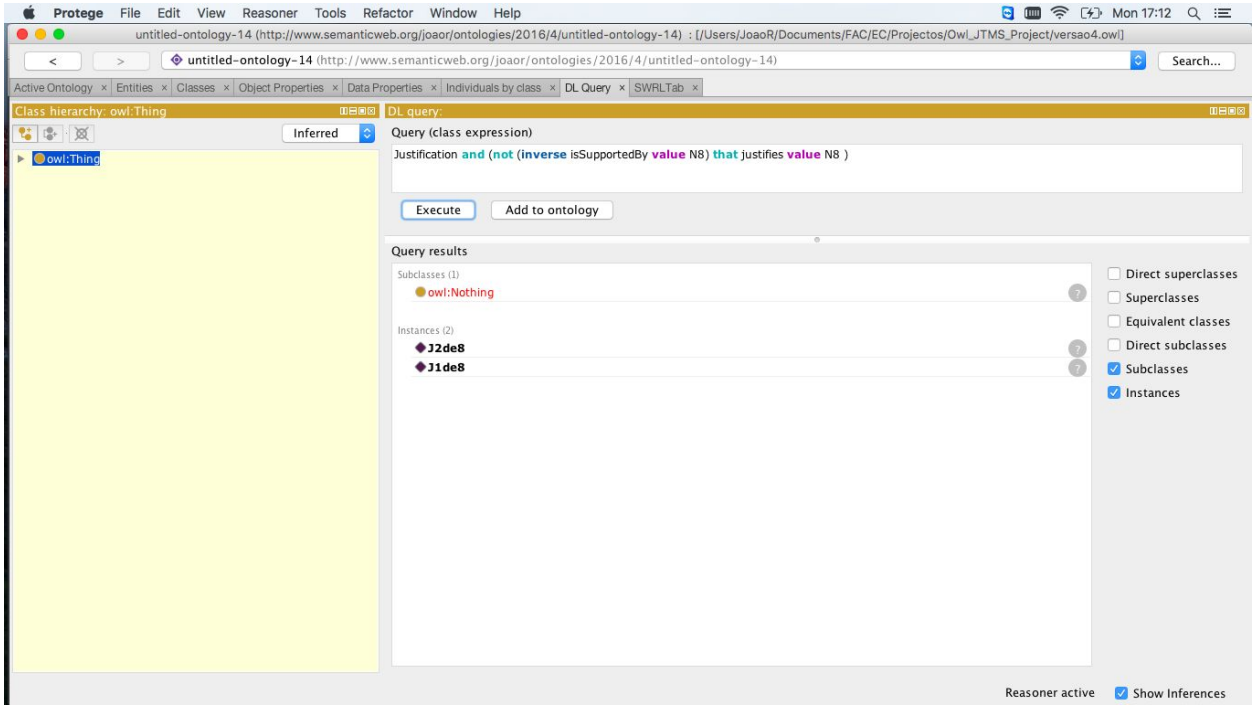
Pergunta 20: Quais as justificações que não são de suporte de N1?

Query: Justification and (not (inverse isSupportedBy value N1) that justifies value N1)



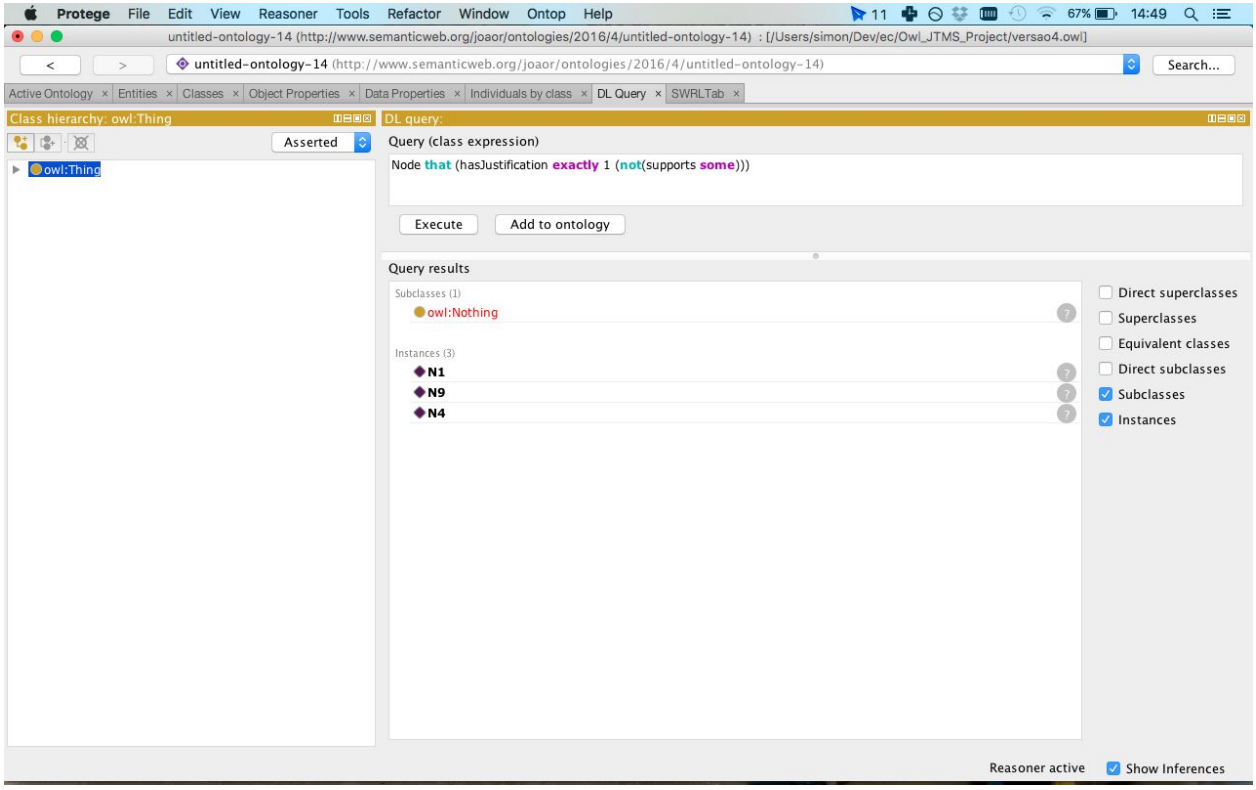
Pergunta 21: Quais as justificações que não são de suporte de N8?

Query: Justification and (not (inverse isSupportedBy value N8) that justifies value N8)



Pergunta 22: Quem tem apenas uma justificação que não é de suporte?

Query: Node that (hasJustification exactly 1 (not (supports some)))



Conclusão

A ontologia definida aceita qualquer tipo de JTMS desde que os inputs fornecidos sejam os necessários para que todas as inferências esperadas possam ser realizadas pelo motor de inferências. Optámos por recorrer a regras SWRL porque dado o modelo por nós desenvolvido, e dado o nosso conhecimento de OWL / Protegé foi a única forma de obter os resultados esperados. Fica apenas definir uma forma automática de inserir as justificações de suporte e respectivos nós de suporte.