

Documentação do exemplo.

Pretende-se com esta documentação exemplificar ao utilizador da rede semântica, que comandos podem ser entrados, quais os efeitos da entrada desses comandos e como tirar o melhor partido possível das instruções actualmente permitidas (fase 2 do desenvolvimento da rede).

De forma a actuar num ambiente o mais elaborado possível, o utilizador deverá aceder à «shell» desenvolvida para o efeito.

Esta «shell» não é mais que um filtro, ao qual todos os seus comandos serão submetidos. Assim; caso instrua a rede semântica para uma operação inválida, a shell encarrega-se de evitar o abortar deslegante do programa, informando-o sobre quais os comandos que a rede semântica está preparada para receber.

No entanto, e restringindo-nos à primeira parte do exemplo, vamos explicar, sequencialmente, o efeito das acções que esse exemplo reporta.

Tudo começa com a invocação da shell da rede semântica através do comando (rs). RS tanto pode ser entendido pelo utilizador como Run Shell ou, simplesmente, Rede Semântica. O que interessa naturalmente reter, é que (rs) permite-lhe aceder à shell.

Todos os comandos posteriormente entrados, já são submetidos à «filragem» da shell. É o caso do comando lista; que pode ser invocado com 3 argumentos - 'nos 'relações ou 'variáveis.

Como se vê, o comando lista, permite mais um argumento do que permitia na primeira fase de desenvolvimento da rede semântica; trata-se do parâmetro 'variáveis.

O efeito duma chamada (lista 'variáveis), é o listar dos identificadores de todas as variáveis cuja criação tenha sido requisitada pelo utilizador através dos comandos adequados. No entanto, para saber a sintaxe do comando lista (e de todos os outros) deve atentar na secção de comandos da documentação do utilizador; neste documento referem-se as acções dos comandos só para um exemplo concreto.

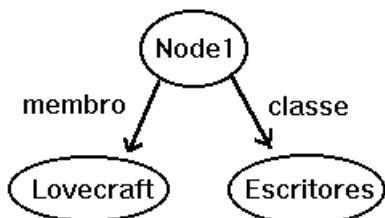
O nosso user exemplo; seja ele o sr. Deacon, iniciou a sua interacção com a shell através da listagem dos nós, relações e variáveis existentes. Como seria de esperar, obteve mensagens alusivas ao facto de todas essas listas não terem ainda elementos.

Sendo um pouco descuidado, o mesmo sr. Deacon, tentou desde logo acrescentar conhecimento à rede através da operação build. Esqueceu-se no entanto de definir a relação membro. Após ler a mensagem de erro que o programa lhe devolveu, Deacon não hesitou em definir duma vez as relações que considerava mais frequentes para o tipo de conhecimento que se propunha capturar; desse modo definiu as relações membro, classe, agente, verbo e objecto.

Já dispondo das relações atrás mencionadas, Deacon procedeu não ao build, mas à asserção do primeiro conceito da sua rede - Deacon capturou o facto de Howard Phillip Lovecraftter sido um escritor.

Para se certificar da criação do nó, Deacon recorre ao comando dump; fica então satisfeito por verificar que o nó requisitado foi, de facto criado!

Graficamente, Deacon provocou a criação do nó:

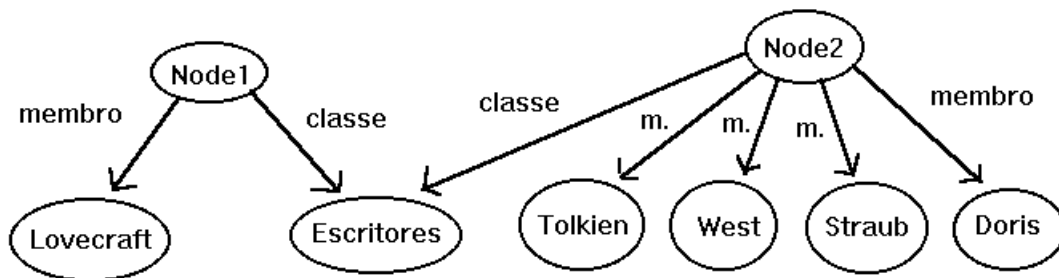


Uma vez criado este nó, Deacon resolveu experimentar o comando descreve. Foi com agrado que viu desfilar a descrição dos 3 nós cuja criação tinha provocado.

Decidido a fazer crescer a rede semântica, Deacon pensou nos seus escritores preferidos e resolveu criar um nó que os assinala-se como membros da mesma classe de Lovecraft. Ainda hesitou em distribuir os escritores por classe mais específicas; como Lovecraft e Peter Straub para uma classe de escritores género fantástico; Doris Lessing e Morris West para género pré-contemporâneo e John Ronald Reuel Tolkien para classe culto da terra e da fantasia, depois, no entanto lembrou-se da fina linha que separa estes géneros e optou por não retroceder.

Seguiu-se uma breve inspecção dos nós criados.

A rede de Deacon, neste momento, estava com o seguinte aspecto gráfico:

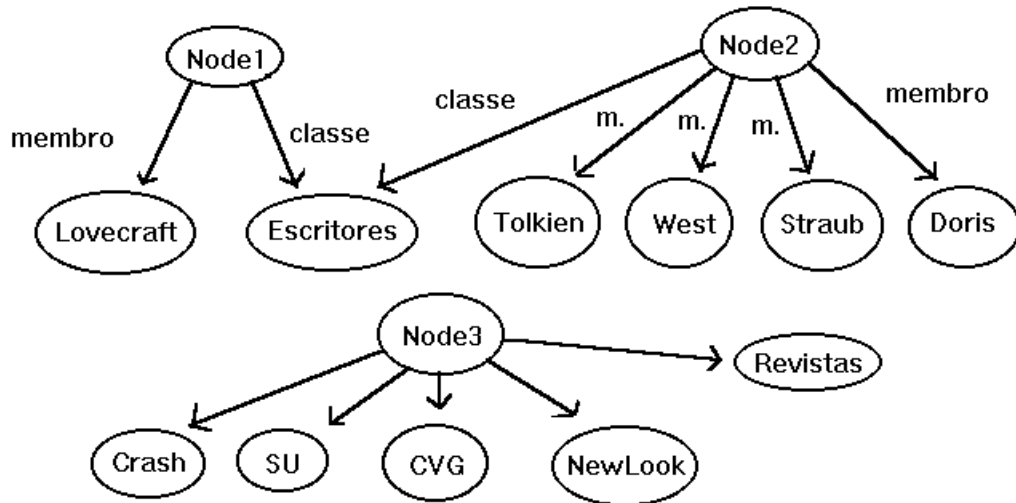


Ainda no campo das letras, Deacon lembrou-se de algumas das revistas que colecionava: a Crash, a Sinclair User, a Computer & Video Games, a NewLook, etc.

Bastavam-lhe essas por agora, afinal só estava a testar a rede semântica.

Fez assim mais uma assercao (tal como poderia ter feito um build), dos conceitos que desejava capturar. Criou (entre outros) o Node3, e procedeu ao respectivo dump; só para confirmar, claro.

Aspecto gráfico da rede do sr. Deacon após a criação do Node3:



Estava chegada a altura de experimentar as novas possibilidades de procura oferecidas pela rede na fase 2 do seu desenvolvimento.

Deacon resolveu então determinar quem eram os membros da classe das revistas.

Fez a chamada (procura 'membro '?quem 'classe 'revistas). Foi-lhe devolvido (NODE3), o que significa que é o Node3 a conter os nós que constituem a resposta à sua pergunta. Conhecedor da sintaxe da função procura, Deacon sabia que havia requisitado a criação duma variável - Quem - e que, para assinalar essa requisição deveria assinalar com ? o nome da variável desejada na chamada a procura; ele tinha feito exactamente isso, estava pois apto a consultar o valor da variável. Para consultar o valor recorreu à função valor?, passando-lhe como argumento o identificador da variável - Quem; ou seja, fez (valor? 'quem), e obteve como resposta (Crash SU CVG NewLook), o que está nitidamente correcto.

Deacon fez de seguida uma acção semelhante para com a classe dos escritores, tendo obtido também a resposta correcta.

Para experimentar as novas funções surgidas pelo aparecimento de variáveis na rede (não confundir com nós variáveis), Deacon recorreu à função lista com o parâmetro 'variáveis, obtendo de seguida o nome do identificador da única variável que tinha solicitado.

Só por curiosidade, Deacon utilizou a seguir a função tira-var, invocada com o parâmetro 'quem, correspondente ao nome da única variável criada, provocando assim o desaparecimento dessa mesma variável.

Relacionado com a procura de respostas a questões sobre o conhecimento suportado na rede, só faltava a Deacon experimentar a função procura-faz, cujo comportamento ele sabia ser exactamente igual ao da função procura, com o acrescento de provocar a criação de nós (tipo hipótese) caso o conceito manifesto na chamada não fosse ainda existente.

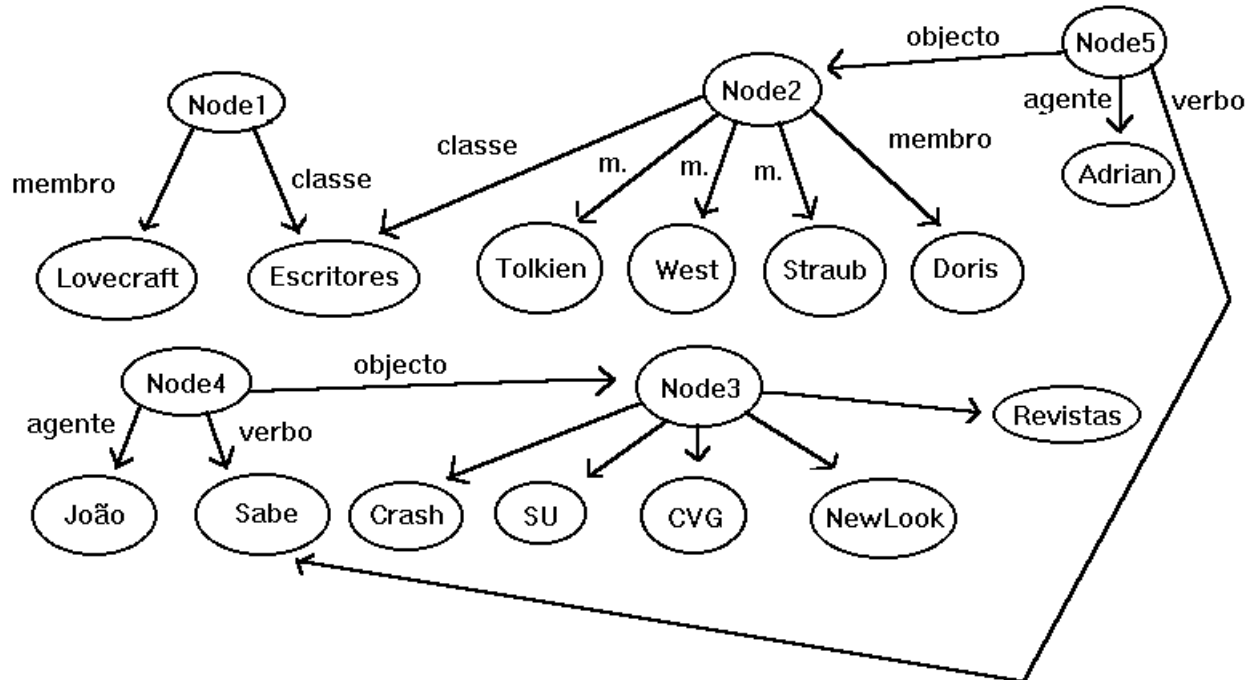
Decidiu-se então a criar um conceito que referi-se o facto do seu amigo João saber que a Crash, a Sinclair User, a CVG e a NewLook são revistas. Criou esse conceito

(Node4) através da função procura-faz.

Fez de seguida uns quantos dumps, só para averiguar que a rede estava a ser construída tal qual era suposto estar.

Estava tudo OK, era altura de complicar ainda mais a rede. Optou por criar um nó que envolvesse o seu adorador Adrian Mole, grande conhecedor de literatura; por isso criou o nó Node5 que referia que Adrian sabia que Lovecraft, Tolkien, Doris, Straub e West eram escritores.

Uma vez criado o Node5, a rede ficou com o seguinte aspecto:



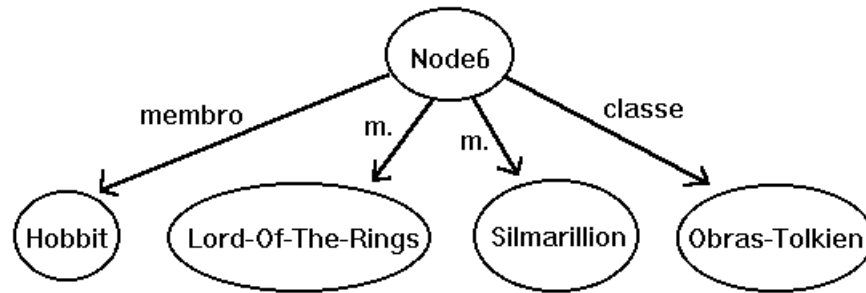
Seguem-se mais uns dumps, para confirmar a solidez do conhecimento suportado pela rede. É também chegada a altura para Deacon fazer uns testes mais arriscados às funções de procura.

Primeiro, por (procura 'membro '?quem), pretende obter todos os nós que são membros de alguma classe. A resposta não falha, e devolve todos os escritores e todas as revistas - precisamente os elementos que na rede estavam atribuídos como sendo membros duma certa classe. E quais as classes que a rede está a suportar ? Uma chamada (procura 'classe '?quais) resolve a questão, devolvendo a resposta na variável de identificador «quais» e de valor consultável pela função valor, através duma chamada do tipo (valor? 'quais).

Os testes às funções de procura sucedem-se, desta feita para saber quem são os agentes. Listam-se depois as variáveis, os nós e as relações criadas com todas estas operações sobre a rede.

Segue-se a criação do Node6, o qual captura conhecimento relativo ao facto de Tolkien ter escrito «The Hobbit», «The Lord of the Rings» e o «Silmarillion».

O Node6:



Após os habituais dumps, Deacon faz a procura:
(procura 'membro '?quem 'classe 'obras-tolkien).

Naturalmente que a variável «quem» fica a reter o nome das 3 obras de Tolkien que a rede está a suportar. Porque não utilizar então o valor da variável «quem» para criar um novo nó - Node7 - que explicita aquilo que está implícito no Node6 ? isto é, um nó que diga que Tolkien escreveu as 3 obras que estão assinaladas como membros da classe Obras-Tolkien ?

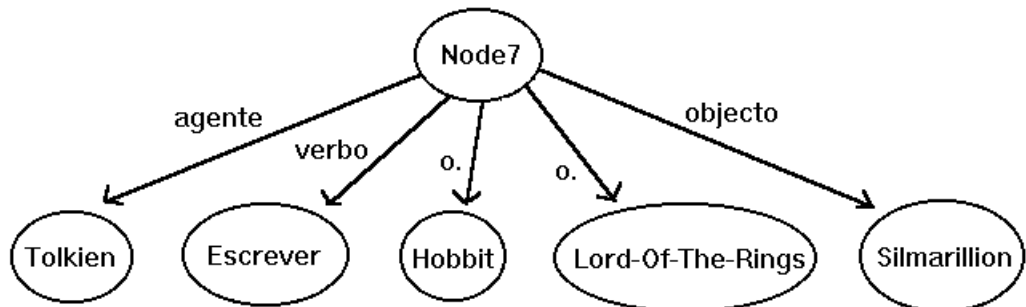
Deacon resolve fazer isso mesmo, passando pois a tirar um considerável proveito das novas facilidades concedidas nesta fase 2 de desenvolvimento da rede semântica. Para fazer o sugerido, bastou-lhe:

(asserção 'agente 'Tolkien 'verbo 'escrever 'objecto (valor? 'quem)).

Atentar no facto de para aceder ao valor da variável «quem» ser necessário chamar a função criada para o efeito - valor?.

Mais uma vez seguem-se uns dumps para confirmar que as relações descendentes e ascendentes, assim como a reutilização de nós se estão a processar de acordo com o previsto.

O Node7:



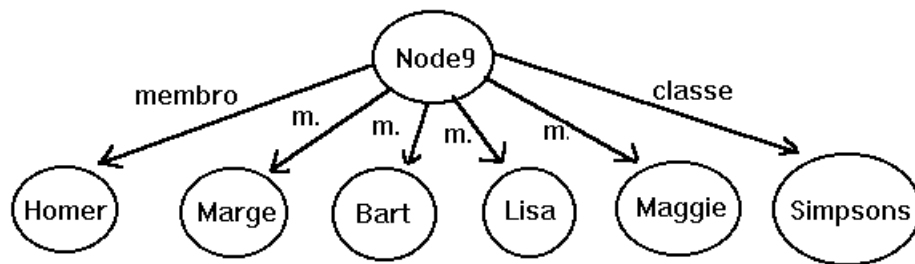
Para testar a função *retira*, Deacon resolve criar um Node8 que mencionava, erradamente e de propósito, o pato Donald como membro da classe dos Simpsons. Deacon invocou então (*retira* 'Donald), solução essa que obviamente não resultou, na

medida em que o nó Donald tinha relações ascendentes para outro nó (o próprio Node8). Não houve então alternativa senão fazer (retira 'Node8).

Voltou-se a fazer uma asserção idêntica, mas desta feita correcta. O nó criado não foi o Node8, mas sim o Node9, isso porque a atribuição de nomes aos nós hipótese ou asserção é controlada por um contador que está sempre a ser incrementado aquando duma operação build ou asserção, não sendo jamais decrementado, funcionando desse modo como um instrumento para saber quantas operações build ou asserção foram feitas desde o início da interacção com a rede. Recordar que, para saber o número de nós suportados num dado momento pela rede, existe a função quantos-nós?.

Deacon, só para confirmar o desaparecimento do Node8, faz (dump 'Node8), obtendo uma mensagem de confirmação da não existência do nó.

O Node9 criado é:



Só para não deixar mal o Donald, Deacon finaliza a sua sessão de interacção com a rede semântica criando um nó semelhante ao Node9, só que desta feita para personagens de Walt Disney. Depois, e para poder recuperar mais tarde a rede agora desenvolvida, Deacon grava-a no ficheiro Rede.Dat, através da operação (grava "Rede.Dat").

Para mais tarde recuperar o ficheiro, bastará escrever (Carrega "Rede.Dat").