Aula 6
Atividade 2

Soar Tutorial 8 – *Semantic Memory*

1) Introdução

Em SOAR, a memória semântica (MS) é um mecanismo que permite a um agente armazenar e recuperar objetos que, em contraste à memória procedural e de trabalho, são permanentes. Esta memória pode ser carregada de um arquivo no início da execução de um agente, exercendo papel de conhecimento pré-existente. Os elementos da MD podem ser completamente independentes das outras memórias e podem ser modificados pelo agente durante a execução do mesmo.

2) Modificando a MS manualmente

No Soar Java Debugger é possível adicionar e visualizar elementos da MS através de comandos. Exemplos são:

a) Adição de elementos:
`smem --add { (elemento 1)... (elemento n) }`
b) Impressão dos elementos
`smem --print`
c) Reinicialização da memória
`smem --init`
d) Impressão de estrutura de grafo
smem --viz

A seguir um exemplo da utilização destes comandos para ilustrar a relação de parentesco entre pessoas de uma família. O texto em azul representa comandos enquanto o texto em vermelho o output do Soar Java Debugger:

```
smem --add {
              (<a> ^name Alice   ^age 60)
              (<b> ^name Bob     ^age 45 ^parent <a>)
              (<c> ^name Charlie ^age 47 ^parent <a>)
              (<d> ^name Dianna  ^age 25 ^parent <b>)
              (<e> ^name Edie    ^age 26 ^parent <b>)
              (<f> ^name Fora    ^age 20 ^parent <c>)
              }

       (nada é exibido)

smem --print

       (@A1 ^age 60 ^name Alice [+1.000])
       (@B1 ^age 45 ^name Bob ^parent @A1 [+2.000])
```

```
(@C3 ^age 47 ^name Charlie ^parent @A1 [+3.000])
(@D1 ^age 25 ^name Dianna ^parent @B1 [+4.000])
(@E2 ^age 26 ^name Edie ^parent @B1 [+5.000])
(@F1 ^age 20 ^name Fora ^parent @C3 [+6.000])
```

```
smem -viz
```

```
digraph smem {

node [ shape = doublecircle ];
A1 [ label="A1\n[+1.000]" ];
B1 [ label="B1\n[+2.000]" ];
C3 [ label="C3\n[+3.000]" ];
D1 [ label="D1\n[+4.000]" ];
E2 [ label="E2\n[+5.000]" ];
F1 [ label="F1\n[+6.000]" ];
node [ shape = plaintext ];
A1_0 [ label = "Alice" ];
A1_1 [ label = "60" ];
B1_0 [ label = "Bob" ];
B1_1 [ label = "45" ];
C3_0 [ label = "Charlie" ];
C3_1 [ label = "47" ];
D1_0 [ label = "Dianna" ];
D1_1 [ label = "25" ];
E2_0 [ label = "Edie" ];
E2_1 [ label = "26" ];
F1_0 [ label = "Fora" ];
F1_1 [ label = "20" ];
A1 -> A1_0 [ label="name" ];
A1 -> A1_1 [ label="age" ];
B1 -> B1_0 [ label="name" ];
B1 -> B1_1 [ label="age" ];
C3 -> C3_0 [ label="name" ];
C3 -> C3_1 [ label="age" ];
D1 -> D1_0 [ label="name" ];
D1 -> D1_1 [ label="age" ];
E2 -> E2_0 [ label="name" ];
E2 -> E2_1 [ label="age" ];
F1 -> F1_0 [ label="name" ];
F1 -> F1_1 [ label="age" ];
B1 -> A1 [ label ="parent" ];
C3 -> A1 [ label ="parent" ];
D1 -> B1 [ label ="parent" ];
E2 -> B1 [ label ="parent" ];
F1 -> C3 [ label ="parent" ];
}
```

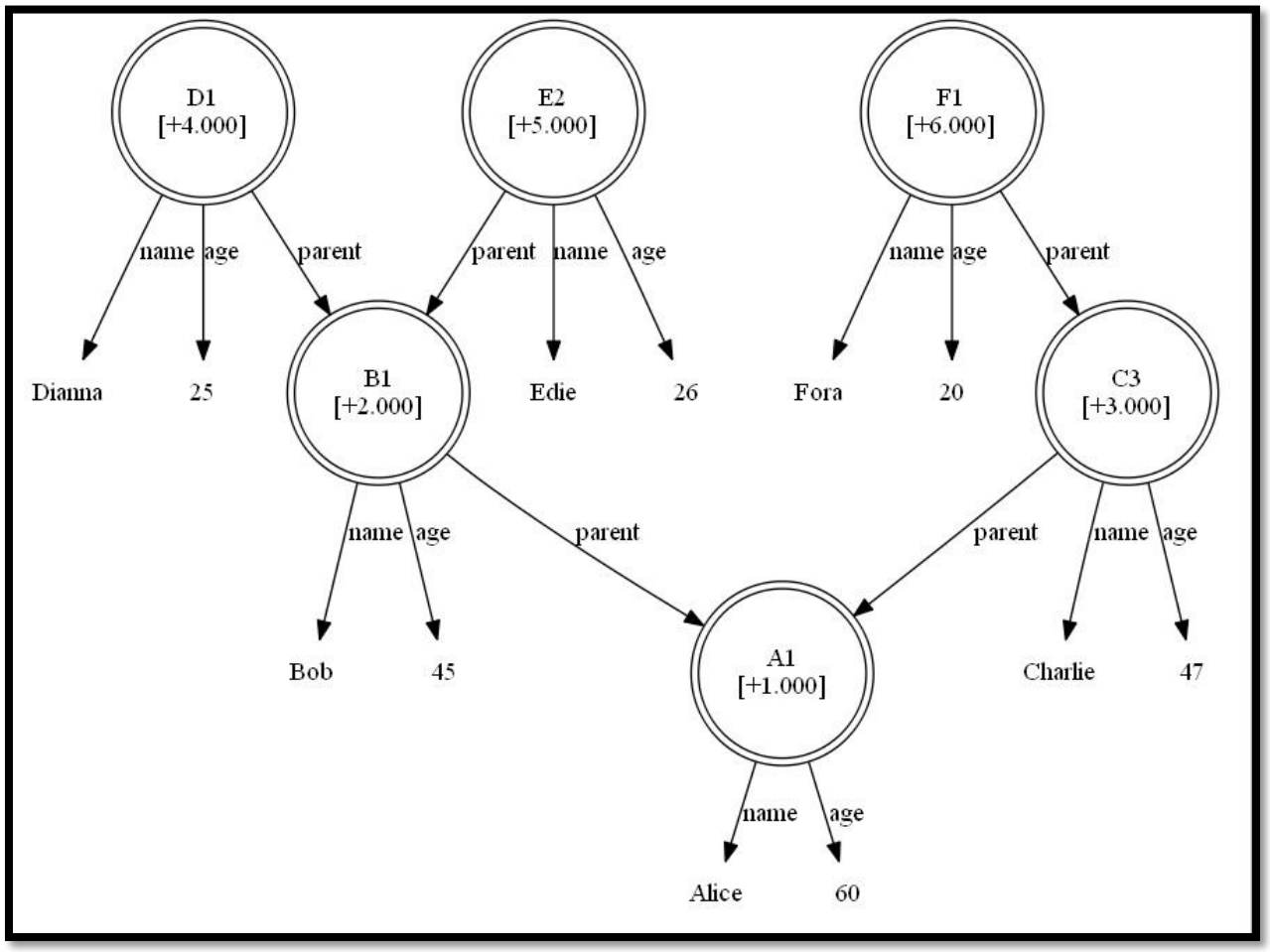
```
smem --init
```

```
0: ==>S: S1
Agent reinitialized.
0 productions excised.
```

```
smem --print
```

(nada é exibido)

Podemos utilizar a saída do comando `smem --viz` como entrada para um programa que saiba desenhar grafos. Utilizando o graphviz (graphviz.org) temos a seguinte visualização:



Esta é claramente uma forma útil de compreender 'o que se passa na mente de um agente' em um determinado momento.

3) Interação do agente com a MS

Existem estruturas na memória de trabalho especialmente reservadas para interação do agente com a memória semântica. Aplicando o comando `print -depth 2 <s>` temos o seguinte resultado:

```
(S1 ^epmem E1 ^io I1 ^reward-link R1 ^smem S2 ^superstate nil ^type state)
(E1 ^command C1 ^present-id 1 ^result R2)
(I1 ^input-link I2 ^output-link I3)
(S2 ^command C2 ^result R3)
```

As estruturas destacadas são justamente aquelas que podem ser utilizadas pelo agente para interação com a **MS**. Ele fornece comandos através de ***command*** enquanto a arquitetura fornece output através de ***result***.

Para ativar este mecanismo é preciso utilizar o seguinte comando:

```
smem --set learning on
```

O tutorial explica a utilização de três comandos: *store*, *query* e *retrieve.* Antes de mais nada, é necessário compreender que os comandos enviados para a arquitetura SOAR não são nada mais que ***augumentações*** do *command-link* (atributo *command* de S2 identificado acima por C2) do objeto que representa a MS (identificado por S2 acima) e que o resultado destes comandos são inseridos pela arquitetura no *result-link* (atributo *result* identificado por R3 acima).

As três *augumentações* que representam comandos explicadas pelo tutorial são:

a) Store*:*

Para armazenar um elemento na MS, augumenta-se o atributo *store* do *command-link* com os identificadores dos elementos a serem armazenados ou atualizados. Ao final da fase onde esta augumentação foi realizada, a arquitetura realiza o armazenamento. Se o identificador for de curta-duração[1], um novo identificador de longa-duração será criado e augumentado pela arquitetura no *result-link* como um atributo ^*success* (um para cada elemento adicionado na MS). Se o identificador for de longa duração o elemento na MS identificado por este identificador será sobrescrito pelo novo.

Por exemplo, se três identificadores <a>, <b> e <c> de curta duração forem inseridos através do comando

```
(<cmd> ^store <a> <b> <c>)
```

Onde `<cmd>` é o *command-link*, o seguinte estado se seguirá ao final da fase.

```
(S2 ^command C2 ^result R3)
    (C2 ^store @B1 ^store @C3 ^store @A1)
    (R3 ^success @B1 ^success @C3 ^success @A1)
```

Onde S2 é o identificador de MS. Podemos ver que tanto o *command-link* quanto o *result-link* possuem agora os identificadores de longa duração que representam, na MS, os elementos armazenados. Modificar estes elementos apenas não altera a MS. É necessário que eles sejam alterados e augumentados no *command-link* com o atributo *store*.

É importante notar que, apesar de vários comandos *store* poderem ser lançados em um mesmo momento, eles não são recursivos para as augumentações do identificador: apenas o primeiro nível de augumentações é copiado para a MS.

b) Retrieve: O comando funciona augumentando-se o *command-link* com o atributo *retrieve* que deve ter como valor o identificador de longa-duração do elemento da MS que se deja recuperar. Se o elemento não existir, um erro será retornado para o *result-link*, enquanto se for o caso contrário o identificador passará a possuir todas as augumentações que o

[1] Identificadores de elementos na MS são chamados identificadores de longa duração, enquanto quaisquer outros de curta-duração. Estes elementos são identificados com um '@'

elemento correspondente possui na MS. Assim como o comando *store*, *retrieve* não é recursivo: o identificador é augumentado apenas com as augumentações de primeiro nível do objeto.

c) ***Query:*** Este comando, também utilizado para recuperar informações da MS, não utiliza identificadores para a busca. Em vez disso, é fornecido um elemento com augumentações quaisquer e será retornado para o *result-link* um elemento da MS que for compatível com o elemento fornecido. Se mais de um elemento for compatível com a busca, um deles é retornado aleatoriamente.

Para ilustrar o funcionamento destes comandos, o tutorial fornece um código fonte SOAR com uma aplicação simples que armazena e recupera elementos da MS. Este código será agora explicado passo a passo.

smem --set learning on	Ativa o mecanismo de MS
sp {propose*init	Proposição do operador de inicialização do agente
(state <s> ^superstate nil	
-^name)	
-->	
(<s> ^operator <op> +)	
(<op> ^name init)	
}	
sp {apply*init	Aplicação da inicialização do agente
(state <s> ^operator.name init	Seja <s> um estado com operador de nome init selecionado
^smem.command <cmd>)	Seja <cmd> o *command-link* do estado
-->	
(<s> ^name friends)	O nome do estado passa a ser friends
(<cmd> ^store <a> <b> <c>)	Augumenta-se o command link com ^store <a>, ^store<b> e ^store <c>
(<a> ^name alice ^friend <b>)	Onde <a> tem nome alice e amigo <b>
(<b> ^name bob ^friend <a>)	Onde <b> tem nome bob e amigo <a>
(<c> ^name charley)	Onde <c> tem nome charley
}	

	Ao final da faze de aplicação deste operador, a arquitetura insere os elementos identificados pelos identificadores de curta duração a, b e c na MS e atribui a eles identificadores @A1, @B1 e @C1 de longa duração.
	
	Estado da MS após aplicação do operador
	
	`(@A1 ^friend @B1 ^name alice [+1.000])`
	`(@B1 ^friend @A1 ^name bob [+2.000])`

	(@C3 ^name charley [+3.000]) Estado do *command-link* e *result-link* após aplicação: (S2 ^command C2 ^result R3) (C2 ^store @B1 ^store @C3 ^store @A1) (@B1 ^friend @A1 ^name bob) (@C3 ^name charley) (@A1 ^friend @B1 ^name alice) (R3 ^success @B1 ^success @C3 ^success @A1)
sp {propose*mod	Proposição do operador mod
(state <s> ^name friends	
^smem.command <cmd>)	
(<cmd> ^store <a> <b> <c>)	Se existe no *command-link* augumentações ^store <a>, ^store<b> e ^store <c> com atributos alice, bob e charley respectivamente
(<a> ^name alice)	
(<b> ^name bob)	
(<c> ^name charley)	
-->	
(<s> ^operator <op> +)	Proponha o operador <op> de nome mod
(<op> ^name mod)	
}	
sp {apply*mod	Aplicação do operador mod
(state <s> ^operator.name mod	Se existe no estado <s> atual um operador de nome mod selecionado
^smem.command <cmd>)	Seja <cmd> o *command-link* deste estado
(<cmd> ^store <a> <b> <c>)	E <a>, <b> e <c> elementos com augumentações name de valor alice, bob e charley respectivamente. Estes elementos serão identificadores de longa duração por causa da aplicação do operador init.
(<a> ^name alice)	
(<b> ^name bob)	
(<c> ^name charley)	
-->	
(<a> ^name alice -)	Remova a augumentação name alice de <a>
(<a> ^name anna	Adicione uma nova augumentação name anna em <a>
^friend <c>)	Adicione uma nova algumentação friend <c> em <a>
(<cmd> ^store <b> -)	Remova as augumentações ^store <b> e ^store <c> do command link
(<cmd> ^store <c> -)	
}	
	Ao final da fase de aplicação deste operador, a arquitetura sobrescreverá na MS o elemento de identificador de longa duração que <a> referencia (no caso, @A1). @B1 e @C1 não serão modificados pois não existem augumentações *store* que os tenham como valor *command-link, como se vê abaixo.* (S2 ^command C2 ^result R3) (C2 ^store @A1) (@A1 ^friend @C3 ^friend @B1 ^name anna) (R3 ^success @A1)

	A MS neste momento se encontra no seguinte estado:
	
	`(@A1 ^friend @B1 @C3 ^name anna [+4.000])`
	`(@B1 ^friend @A1 ^name bob [+2.000])`
	`(@C3 ^name charley [+3.000])`

sp {propose*ncb-retrieval	Proposição do operador ncb-retrieval que irá buscar na MS um elemento dado seu identificador de longa duração apenas.
(state <s> ^name friends	
^smem.command <cmd>)	
(<cmd> ^store <a>)	Seja <a> o valor de uma augumentação do atributo ^store do *command-link*
(<a> ^name anna	Onde <a> tem nome anna
^friend <f>)	Seja <f> um valor de um atributo *friend* de <a>
-->	
(<s> ^operator <op> + =)	Proponha o operador <op>
(<op> ^name ncb-retrieval	De nome ncb-retrieval
^friend <f>)	Com atributo friend <f>
}	
sp {apply*ncb-retrieval*retrieve	Aplicação da ação retrieve do operador nbc-retrieval que adiciona no *command-link* uma augumentaçãp retrieve para o atributo friend do operador
(state <s> ^operator <op>	Seja <f> o valor do atributo friend atribuído ao operador
^smem.command <cmd>)	
(<op> ^name ncb-retrieval	
^friend <f>)	
(<cmd> ^store <a>)	Seja <a> o valor de uma augumentação ^store do *command-link*
-->	
(<cmd> ^store <a> -	Remove a augumentação do *command-link*
^retrieve <f>)	Insere uma augumentação para ^*retrieve* <f> (<f> é um identificador de longa-duração que será um dos amigos de <a> (@B1 ou @C3), escolhido aleatoriamente pela arquitetura.
}	
sp {apply*ncb-retrieval*clean	Aplicação da ação clean do operador nbc-retrieval que remove todas as augumentações do atributo friend do operador. Esta ação foi tomada para demonstrar que basta o identificador de longa duração para resgatar da MS o elemento desejado.
(state <s> ^operator <op>	Seja <f> o valor do atributo friend atribuído ao operador
^smem.command <cmd>)	
(<op> ^name ncb-retrieval	

^friend <f>)	
(<f> ^<attr> <val>)	Seja <attr> o atributo de uma augumentação de <f> e <val> seu valor
-->	
(<f> ^<attr> <val> -)	Remove a augumentação ^<attr> <val>
}	

	No final da fase de aplicação deste operador, o comando store do operador anterior foi removido do *command-link* e foi adicionado um comando retrieve para um dos amigos <f> de @A1. Como nenhum comando *store* ocorreu, a MS não foi alterada. Os estados do *command-link* e *result-link* neste momento são:
	`(S2 ^command C2 ^result R3)`
	`  (C2 ^retrieve @C3)`
	Durante a fase de output, a busca na MS pelo elemento @C3 (escolhido aleatoriamente) será realizada. A seguir, o estado do command-link e result-link após a fase de output.
	`(S2 ^command C2 ^result R3)`
	`  (C2 ^retrieve @C3)`
	`    (@C3 ^name charley)`
	`  (R3 ^retrieved @C3 ^success @C3)`
	Podemos ver que o elemento @C3 foi adicionado na memória de trabalho com atributos correspondentes ao que se encontra na MS e seu identificador se encontra tanto no *command-link* quanto no *result-link*

sp {propose*cb-retrieval	Proposição do operador cb-retrieval que irá buscar na MS um elemento que seja compatível com um parâmetro, sem utilizar o identificador de longa-duração
(state <s> ^name friends	
^smem.command <cmd>)	
(<cmd> ^retrieve)	Se algum comando *retrieve* foi feito
-->	
(<s> ^operator <op> + =)	Propõe o operador cb-retrieval
(<op> ^name cb-retrieval)	
}	
sp {apply*cb-retrieval	Aplicação do operador cb-retrieval
(state <s> ^operator <op>	
^smem.command <cmd>)	
(<op> ^name cb-retrieval)	
(<cmd> ^retrieve <lti>)	Seja <lti> o elemento que foi recuperado pela operação retrieval
-->	

(<cmd> ^retrieve <lti> -	Remove o comando retrieve do *command-link*
^query <cue>)	Adiciona a augumentaçãp ^query <cue> ao command-link
(<cue> ^name <any-name>	Onde <cue> é um elemento com um atributo name de valor qualquer (<any-name> não existe no contexto atual)
^friend <lti>)	E um atributo friend com o valor <lti>, o identificador de longa duração recuperado na operação anterior
}	

	Ao final da fase de aplicação deste operador, foi adicionado o comando query no command-link. Os valores no *result-link* ainda são os da operação anterior:
	
	`(S2 ^command C2 ^result R3)`
	`  (C2 ^query C4)`
	`  (C4 ^friend @C3 ^name A2)`
	`  (R3 ^retrieved @C3 ^success @C3)`
	`  (@C3 ^name charley)`
	
	A MS não foi alterada desde a aplicação desde a aplicação do operador mod, e se encontra no seguinte estado:
	
	`(@A1 ^friend @B1 @C3 ^name anna [+4.000])`
	`(@B1 ^friend @A1 ^name bob [+2.000])`
	`(@C3 ^name charley [+3.000])`
	
	Durante a fase de output, a arquitetura fará a busca na MS por um elemento que possui atributo amigo valendo @C3 e com augumentação name qualquer. No momento, o único elemento que é compatível com esta configuração é @B1, que será o resultado desta busca, como podemos ver pelo estado do *result-link* ao final da fase de output.
	
	`(S2 ^command C2 ^result R3)`
	`  (C2 ^query C4)`
	`  (C4 ^friend @C3 ^name A2)`
	`  (R3 ^retrieved @A1 ^success C4)`
	`  (@A1 ^friend @B1 ^friend @C3 ^name anna)`

sp {done	Elaboração done
(state <s> ^smem <smem>)	Se houver um comando query <q> no command-link que já retornou algum resultado
(<smem> ^command.query <q>	
^result.<status> <q>)	
-->	
(halt)	Pare a execução do agente.
}	

4) Conclusão

Este tutorial teve como objetivo explicar e ilustrar o funcionamento da interação de um agente com a Memória Semântica. Foram explicados os comandos *store, retrieve* e *query* e foi utilizado um exemplo simples de como utilizar o command-link e result-link para armazenar e recuperar elementos da memória semântica.