

Sistemas de raciocínio monotónico

- Um mecanismo de raciocínio é designado monotónico se deriva novos factos a partir de factos conhecidos
- Nesses sistemas, o número de factos aumenta monotonicamente ao longo do tempo
- Contudo, esses sistemas não são adequados para situações onde:
 - o conhecimento é incompleto
 - o conhecimento é impreciso

Sistemas de Manutenção da Verdade

1

Sistemas de raciocínio monotónico

Exemplo:

- Consideremos uma relação de herança:
Se X é um cão, Então X é um mamífero.
 - Se adicionarmos (à memória de trabalho) o facto:
Pluto é um cão.
 - Então também o seguinte facto será adicionado:
Pluto é um mamífero.
 - Mais tarde, se ficarmos a saber que Pluto não é um cão o que fazer com o facto Pluto é um mamífero? E o que fazer com outros factos derivados?
- A situação aqui representada corresponde a uma situação de não monotonia.

Sistemas de Manutenção da Verdade

2

Sistemas de raciocínio não-monotónico

Motivação

- Muitos dos problemas do mundo real não podem ser completamente especificados – o Motor de Inferência (MI) tem que gerar conclusões baseado em informação incompleta
- Tipicamente a assumpção que está na base de tais conclusões consiste em considerar que um facto X é verdadeiro perante a ausência de evidência do contrário – isto é conhecido como Assumpção do Mundo Fechado (AMF)
- A AMF ajuda-nos a limitar o espaço de pesquisa relevante, assumindo apenas uma dada escolha ou ignorando as restantes
- O mecanismo de raciocínio que utiliza esta assumpção é designado de raciocínio por defeito ou raciocínio não monotónico

Sistemas de Manutenção da Verdade

3

Sistemas de raciocínio não-monotónico

Identificação das causas de inconsistência

A existência de inconsistências entre crenças numa Base de Conhecimento é provável, especialmente se o Motor de Inferência gera as suas conclusões com base em informação suficiente – as razões mais comuns para a existência de inconsistências ou outras anomalias são as seguintes:

- Dados errados - Exemplo: “A temperatura exterior é de 320 °C.”
- Restrições impossíveis - Exemplo: Casa grande, Casa barata e Casa com boa qualidade de construção
- Inferência não-monótona – O MI é forçado a admitir uma conclusão devido a falta de informação ou falta de tempo processamento para derivar uma conclusão
- Contradições devidas a dados inconsistentes, conclusões que contrariam dados existentes ou assumpções inconsistentes
- Dados dinâmicos – quando o domínio evolui, o novo estado do domínio pode ser consideravelmente diferente do anterior, tornando inválidas as inferências produzidas durante o estado anterior

Sistemas de Manutenção da Verdade

4

Sistemas de raciocínio não-monotónico

- Uma forma de ultrapassar a situação descrita no exemplo (pág. 2) é usar crenças (ou assumpções), isto é, asserções hipotéticas sobre o domínio, as quais não são completamente suportadas por evidências, mas não estão em contradição com o que já é conhecido
- A partir de uma crença, é possível derivar novos factos mas, se surge uma contradição, o sistema terá que ser capaz de a retirar e de executar o conjunto de ações necessárias para manter o conhecimento coerente
- Um sistema de raciocínio baseado em crenças e apto a gerir contradições é denominado não-monotónico
- Os TMS são exemplos práticos de sistemas não-monotónicos

Sistemas de Manutenção da Verdade


5

Sistemas de Manutenção da Verdade

Truth Maintenance Systems

- Um TMS pode ser visto como um processo de representação de conhecimento para representar as crenças do sistemas, assim como as suas dependências

O nome **manutenção de verdade** deve-se à capacidade destes sistemas em **restaurar** a consistência do conhecimento



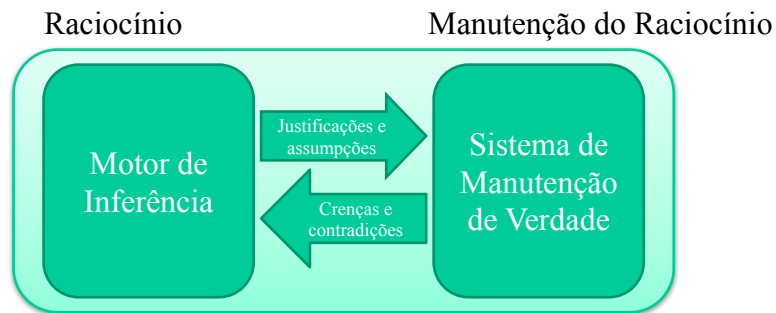
Manutenção do raciocínio

- Um TMS é basicamente um motor de inferência que aplica regras para conduzir as escolhas acerca daquilo em que deve acreditar

Sistemas de Manutenção da Verdade

6

Sistemas de Manutenção da Verdade



- O **Motor de Inferência** explora alternativas, faz escolhas e examina as consequências dessas escolhas. Se é detetada uma contradição durante este processo, o TMS elimina-a revendo a **Memória de Trabalho**

Sistemas de Manutenção da Verdade

7

Sistemas de Manutenção da Verdade

“TRUTH MAINTENANCE SYSTEMS (TMS)“

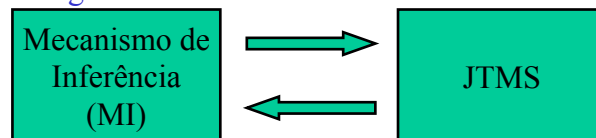
- Funções principais dos TMS (ou sistemas de revisão de crenças):
 - guardar inferências
 - permitir o raciocínio baseado em assumpções
 - gerir inconsistências
- Principais dificuldades existentes nos TMS:
 - representação das crenças e das suas interdependências
 - identificação das crenças responsáveis por contradições
 - remoção de crenças (*backtracking*)

Sistemas de Manutenção da Verdade

8

TMS baseados em justificações

- Justification-Based TMS (JTMS, Doyle-1979)
 - Trata-se de uma abordagem simples de TMS que permite analisar as **consequências do conjunto actual de assumpções**; é capaz de justificar a verdade das suas crenças através da análise de dependências entre as inferências produzidas pelo MI
- Arquitectura genérica de um KBS incluindo um JTMS



Mecanismo de Inferência → resolve problemas num domínio, possivelmente usando assumpções
JTMS → gere a consistência da inferência produzida pelo MI

9

Sistemas de Manutenção da Verdade

TMS baseados em justificações

Ligação MI ↔ JTMS

- O motor de inferência fornece:
 - asserções (ou crenças) acerca do domínio
 - justificações para as crenças deduzidas
- O JTMS fornece:
 - contextos, isto é, informação acerca das crenças actuais, justificações, existência e causas de inconsistências

10

Sistemas de Manutenção da Verdade

TMS baseados em justificações

- Nó 1: If X is a bug and X is new then it has to be fixed.
- Nó 2: Y is a bug and Y is new.
- Nó 3: Z is a bug and Z is closed.
- Nó 4: Y has to be fixed. (Outcome)

Justificação do Nó 4: Nó 1 + Nó 2

Sistemas de Manutenção da Verdade

11

Funcionalidade principal de um JTMS

Manutenção da consistência das crenças

- O JTMS opera sobre um conjunto de nós (representando crenças) e de justificações (informação que suporta as crenças)
- As justificações de um nó genérico X são representadas através da especificação dos nós que implicam X
- Durante este processo, o MI fornece ao JTMS informação que permite:
 1. a criação de novos nós
 2. a actualização (adição ou remoção) de justificações
 3. a identificação da responsabilidade das contradições
- As contradições são tratadas através da reformulação das dependências directas

Sistemas de Manutenção da Verdade

12

Labelling

- A todos os nós é atribuído um estado:
 - IN (presentemente admissível ou aceite como verdadeiro) ou
 - OUT (presentemente não admissível ou não aceite como verdadeiro).
- O JTMS efectua um processo de *labelling* cujo objectivo é identificar o estado de cada nó. Quando todos os nós têm um estado coerente, então o conjunto de nós diz-se relaxado
- No final do processo de *labelling* o MI é forçado a trabalhar apenas com os nós etiquetados como IN

Sistemas de Manutenção da Verdade

13

Justificações

- As Justificações guardam as dependências entre os nós “justificados” e os seus antecedentes
- Uma justificação contém duas listas (*support lists*):
 - *inlist* e *outlist* – para descrever quais os nós que devem estar *in* e quais devem estar *out* para que um nó justificado seja considerado verdadeiro
- As premissas correspondem a proposições sempre verdadeiras – as justificações das premissas contêm *inlist* e *outlist* vazias – este tipo de nós estará sempre *in*, independentemente do estado dos outros nós
- As justificações das *assumpções* contêm *outlists* não vazias – como mencionado anteriormente, uma *assumpção* depende tanto da informação que suporta directamente a sua veracidade, assim como da ausência de evidências contra ela

Sistemas de Manutenção da Verdade

14

Operação do JTMS

- As justificações são usadas com 2 propósitos:
 1. Actualizar o conjunto de crenças sempre que o estado de admissibilidade de um nó é alterado. As justificações são usadas para encontrar todos os nós afectados, sendo os seus estados de admissibilidade reexaminados. Como resultado, alguns desses nós passarão ao estado *in* e outros ao estado *out*
 2. Gerir a ocorrência de contradições. Quando uma contradição é descoberta pelo MI, é adicionada uma justificação relativa a essa contradição e é invocado um sistema de reformulação das dependências directas (implementado pelo JTMS). Este sistema procura na rede de dependências (usando justificações e nós) as *assumpções* responsáveis pela contradição. Uma destas *assumpções* é seleccionada como responsável pela contradição e um dos nós da sua *outlist* é justificado, removendo-se assim a contradição.

Sistemas de Manutenção da Verdade

15

Exemplo (1)

Pretende-se avaliar o desempenho de um robot-humanóide com comportamento inteligente numa situação de férias.

Situação:

Está um dia de sol e o robot decide ir nadar.

Contudo, surgem nuvens escuras e começa a chover.

O robot recolhe os seus pertences e desloca-se para o seu quarto.

No quarto, o robot pega no seu livro favorito e começa a ler.

Passado algum tempo, o robot sente-se cansado, não conseguindo manter os olhos abertos.

Entretanto é noite e o robot decide ir dormir.

Sistemas de Manutenção da Verdade

16

Exemplo (2)

Situação inicial

id proposição	proposição	justificação
A	temperatura ≥ 25	[[], [B]]
B	temperatura < 25	
C	não chove	[[], [D]]
D	chove	
E	dia	[[], [F]]
F	noite	

A, C e E
estão *in*

Através da regra

Se temperatura ≥ 25 e não chove Então tempo agradável
o MI conclui que o tempo está agradável, criando o nó G.
A justificação deste nó é enviada para o JTMS, o qual classifica
este nó como *in*

Sistemas de Manutenção da Verdade

17

Exemplo (3)

id proposição	proposição	justificação
G	tempo agradável	[[A, C], []]

Através da regra

Se 'tempo agradável' e é 'dia' Então 'nadar'
o MI conclui que o sujeito vai nadar, criando o nó H

id proposição	proposição	justificação
H	nadar	[[E, G], []]

Quando começa a chover, a situação contradiz a assumpção C (não chove). Para lidar com esta situação, o MI envia para o JTMS uma justificação referente à contradição, representada pelo nó I

id proposição	proposição	justificação
I	contradição	[[C], []]

Sistemas de Manutenção da Verdade

18

Exemplo (4)

O sistema de reformulação de dependências directas (implementado pelo JTMS) é invocado e encontra a assumpção C como causa da contradição. É então usada a premissa X para remover a contradição:

id proposição	proposição	justificação
X	premissa x	[[], []]

O sistema de reformulação de dependências directas selecciona a assumpção C como causa da contradição e justifica o nó D, o antecedente incluído na lista *out* de C:

id proposição	proposição	justificação
D	chove	[[X], []]

Como consequência, o JTMS marca A, D e E *in* e marca B, C, F, G, H e I *out*.

Sistemas de Manutenção da Verdade

19

Exemplo (5)

Através da regra

Se ‘chove’ e é ‘dia’ Então ‘apropriado para ler’

o MI conclui que é apropriado ler. O MI envia para o JTMS uma justificação para o nó J:

id proposição	proposição	justificação
J	apropriado para ler	[[D, E], []]

Para descrever a situação de que o sujeito ficou cansado, o MI deve introduzir uma nova contradição, atribuindo uma justificação ao nó K: (Se ‘cansado’ Então não ‘apropriado para ler’)

id proposição	proposição	justificação
K	contradição	[[J], []]

Sistemas de Manutenção da Verdade

20

Exemplo (6)

O sistema de reformulação de dependências directas encontra a assumpção E como causa da contradição, examinando os antecedentes de J (D e E). O nó X é sempre verdadeiro. Assim, a única assumpção de que J depende é E. A assumpção E é seleccionada como causa da contradição e F é justificado:

id proposição	proposição	justificação
F	noite	[[X], []]

Como consequência, o JTMS marca os nós A, D e F *in* e marca os nós B, C, E, G, H, I, J e K como *out*.

Finalmente, o MI conclui que está na altura do sujeito ir dormir, atribuindo a justificação ao nó L: (Se é 'noite' Então 'dormir')

id proposição	proposição	justificação
L	dormir	[[F], []]

Sistemas de Manutenção da Verdade

21

JTMS – Conclusão

- É adequado para situações em que é necessária apenas uma solução para o problema de gestão da coerência/consistência do conhecimento
- Apenas pode explorar um conjunto de crenças de cada vez (contexto) – a alteração do conjunto de crenças/assumpções apenas é efectuada após a detecção de uma contradição
- Não permite comparar dois conjuntos de crenças em simultâneo
- O processo de identificação de assumpções responsáveis por contradições pode ser complexo

Sistemas de Manutenção da Verdade

22

Outros TMS

- TMS baseado em assumções (ATMS, DeKleer-1986)
 - Raciocínio hipotético – são consideradas diferentes conjuntos de assumções de modo a se considerarem diferentes contextos. Ou seja, é produzido raciocínio acerca de mundos alternativos, independentemente da sua semelhança com o mundo actual
 - Permite manter em simultâneo o raciocínio acerca de diferentes conjuntos de assumções, possivelmente incompatíveis – são mantidas em paralelo linhas de inferência alternativas
 - A diferença principal entre JTMS e ATMS reside na forma de tratamento de contextos. Enquanto que num JTMS existe sempre um só contexto em que os nós podem estar *in* ou *out*, num ATMS existem vários contextos em simultâneo

Sistemas de Manutenção da Verdade

23

Outros TMS

- ATMS (continuação)
 - Nó 1: **If X is a bug and X is new then it has to be fixed.**
 - Nó 2: **Y is a bug and Y is new.**
 - Nó 3: **Z is a bug and Z is closed.**

Justificações e Assumpção:

- Nó 4: **Y is not fixed. (Resultado)**
- Nó 5: **Z has to be fixed. (Crença)**

Ambiente Consistente: **Nó 4: Nó 1 + Nó 2**

Ambiente Inconsistente: **Nó 5: Nó 1 + ~ Nó 3**

Sistemas de Manutenção da Verdade

24

Outros TMS

- TMS baseado em lógica (LTMS, McAllester-1980)
 - Ao contrário dos métodos anteriores, o LTMS não se baseia em justificações para manter as dependências entre nós. O LTMS usa cláusulas lógicas para especificar restrições entre estados dos nós.
 - Mais poderoso do que o JTMS uma vez que reconhece a semântica proposicional das afirmações – adequado para raciocínio proposicional em tempo real
 - (TRUE) Nó 1: X is a bug and X is new.
 - (FALSE) Nó 2: Y is a bug and Y is closed.
 - (TRUE) Nó 3: Z is a bug.
 - O LTMS tenta identificar como TRUE ou FALSE os nós etiquetados como UNKNOWN
 - (UNKNOWN) Nó 4: If Fred fix X then Fred fix Z.
 - (UNKNOWN) Nó 5: Z is closed.

Sistemas de Manutenção da Verdade

25

Truth Maintenance in Drools

- Two kinds of fact insertion:
 - Stated insertion
 - insert(<Fact>)
 - Logical insertion
 - insertLogical(<Fact>)
 - Facts inserted using insertLogical will remain in the session until the LHS of the rule that inserted it becomes false (or you explicitly retract them using retract())

Truth Maintenance Systems

26

Truth Maintenance in Drools

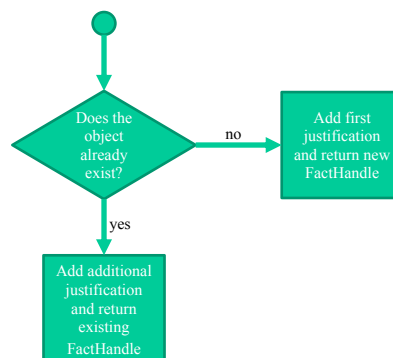
- When we logically insert an object during a RHS execution we are said to justify it, and it is considered to be justified by the firing rule
- For each logical insertion there can only be one equal object, and each subsequent equal logical insertion increases the justification counter for this logical assertion
- A justification is removed by the LHS of the creating rule becoming untrue, and the counter is decreased accordingly
- As soon as we have no more justifications the logical object is automatically retracted

Sistemas de Manutenção da Verdade

27

Truth Maintenance in Drools

- Logical insertion



Truth Maintenance Systems

28

TMS in Drools - Example

```
rule "Infer Child" when
    $p : Person( age < 16 )
then
    insertLogical( new IsChild( $p ) )
end
rule "Infer Adult" when
    $p : Person( age >= 16 )
then
    insertLogical( new IsAdult( $p ) )
end
rule "Issue Child Bus Pass" when
    $p : Person( )
    IsChild( person == $p )
then
    insertLogical(new ChildBusPass( $p ) );
end
rule "Issue Adult Bus Pass" when
    $p : Person( age >= 16 )
    IsAdult( person == $p )
then
    insertLogical(new AdultBusPass( $p ) );
end
```

When a person changes from being 15 to 16, not only is the IsChild fact automatically retracted, so is the person's ChildBusPass fact

Sistemas de Manutenção da Verdade

29

TMS in Drools - Requirement

- For Truth Maintenance (and logical assertions) to work at all, your Fact objects (which may be JavaBeans) must override **equals** and **hashCode** methods (from java.lang.Object) correctly
- As the truth maintenance system needs to know when two different physical objects are equal in value, *both* **equals** and **hashCode** must be overridden correctly, as per the Java standard
- Two objects are equal if and only if their **equals** methods return true for each other and if their **hashCode** methods return the same values

Sistemas de Manutenção da Verdade

30