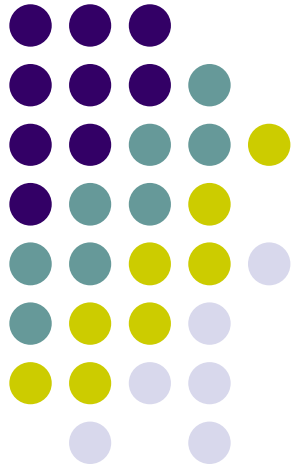
Sistemas de Representação e Raciocínio
Parte 3

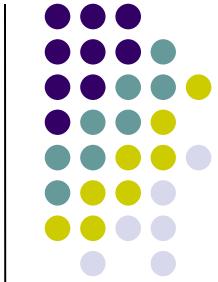
Introdução à Inteligência Artificial

Profa. Josiane

Baseado no material de David Poole,
Alan Mackworth e Randy Goebel
Abril/2007

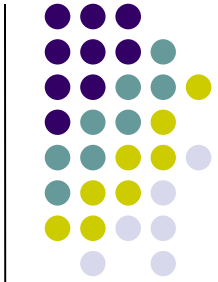


Retomando da aula passada...
Modelos e consequências lógicas



- Uma **BC é verdadeira em uma interpretação I** se, e somente se **toda** cláusula em BC é verdadeira em I
- Um **modelo** de um conjunto de cláusulas é uma interpretação na qual **todas** as cláusulas são verdade
- Seja a BC é um conjunto de cláusulas e g é uma conjunção de átomos
  - g é um **consequência lógica** da BC, escrita como BC |= g, se g é verdade **para todo modelo da BC**
- Isto é, BC |= g se não existe nenhuma interpretação na qual a BC é verdadeira e g é falsa

Modelos – Exemplo



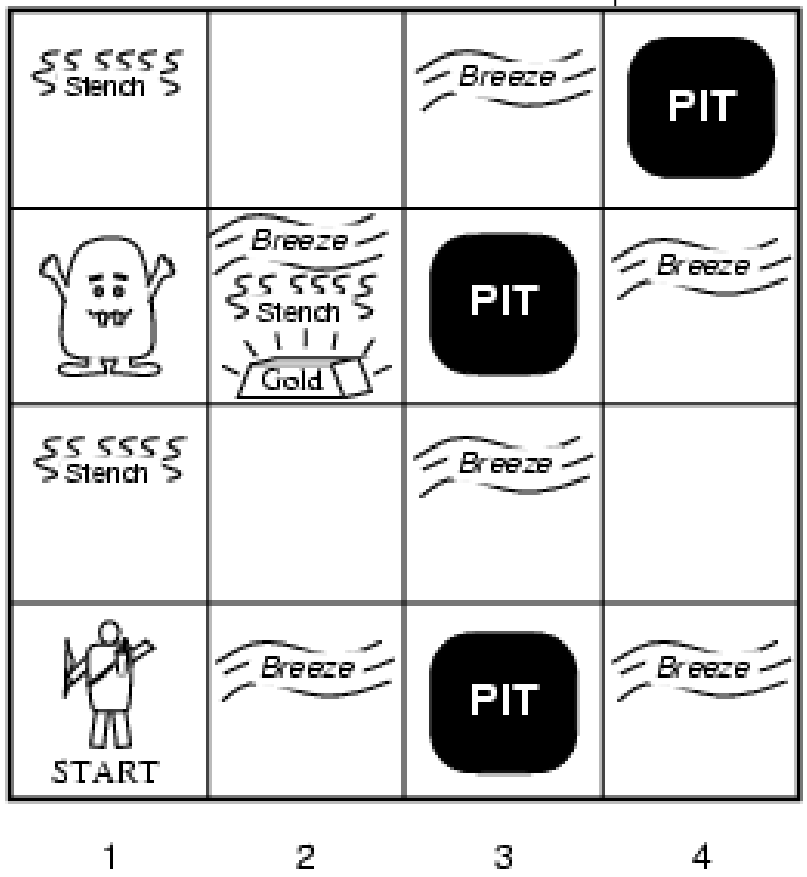
$$KB = \begin{cases} p \leftarrow q. \\ q. \\ r \leftarrow s. \end{cases}$$

	$\pi(p)$	$\pi(q)$	$\pi(r)$	$\pi(s)$	
I_1	*TRUE*	*TRUE*	*TRUE*	*TRUE*	is a model of KB
I_2	*FALSE*	*FALSE*	*FALSE*	*FALSE*	not a model of KB
I_3	*TRUE*	*TRUE*	*FALSE*	*FALSE*	is a model of KB
I_4	*TRUE*	*TRUE*	*TRUE*	*FALSE*	is a model of KB
I_5	*TRUE*	*TRUE*	*FALSE*	*TRUE*	not a model of KB

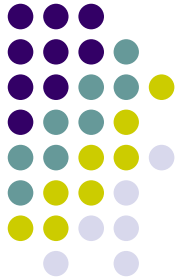
$KB \models p$, $KB \models q$, $KB \not\models r$, $KB \not\models s$

Modelos – Exemplo
O Mundo do Wumpus

- Wumpus e fedor nos adjacentes
- Poços e brisa nos adjacentes
- Ouro e brilho no local
- Atirar mata o Wumpus se o agentes estiver virado para ele
- O agente só tem uma flecha
- Agente morre se entrar em uma sala com um poço ou o Wumpus vivo



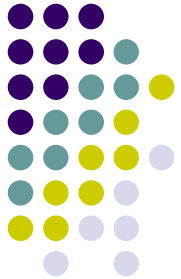
Modelos – Exemplo
O Mundo do Wumpus



- Medida de desempenho
  - Pegar o ouro = +1000
  - Cair em poço ou ser devorado = -1000
  - Executar ação = -1
  - Usar a flecha = -10

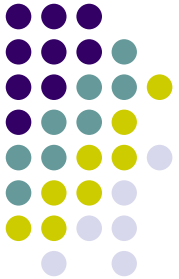
- Ambiente
  - Agente começa em [1,1], voltado para a direita
  - Posições do ouro e do Wumpus são aleatórias
  - Cada sala pode ter um poço com probb. 0,2

Modelos – Exemplo
O Mundo do Wumpus



- Atuadores
  - Agente pode se mover para frente, ou a esquerda e a direita 90º
  - Mover-se para frente não tem efeito se houver uma parede
  - Ação agarrar – pega um objeto que está no mesmo quadrado que o agente
  - Ação atirar – atira a flecha em linha reta diante do agente
    - A flecha irá parar somente quando atingir o Wumpus ou a parede

Modelos – Exemplo
O Mundo do Wumpus



- Sensores: fornecem um único bit de informação
  - Fedor – quadrado com o Wumpus e adjs. a ele
  - Brisa – quadrados adjs. a um poço
  - Brilho – quadrado do ouro
  - Impacto – quando caminhar para uma parede
  - Grito – do Wumpus quando morre
  - Exemplo: [Fedor, Brisa, Nada , Nada , Nada]

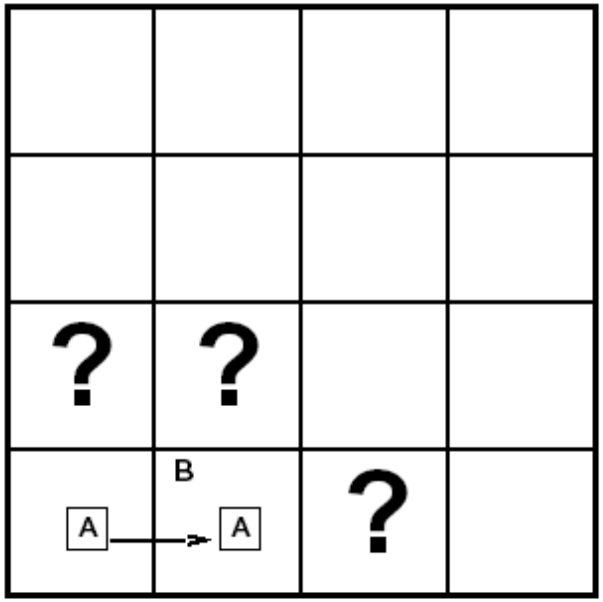
- Principal dificuldade – ignorância inicial da configuração do ambiente – exige exploração e raciocínio

Modelos – Exemplo
O Mundo do Wumpus

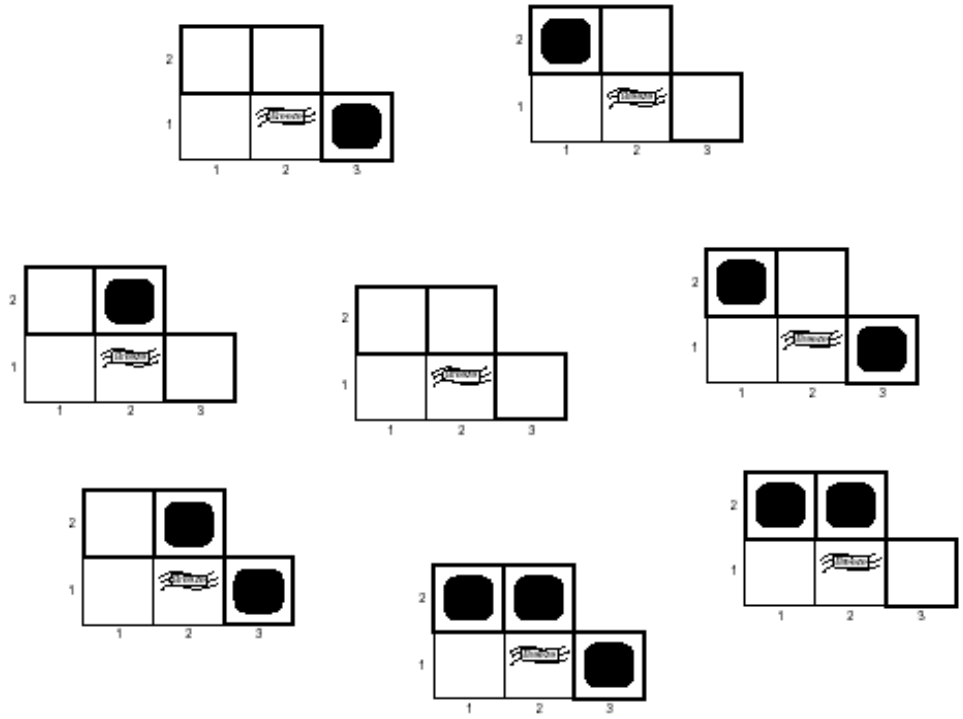
- Inicialmente a BC contém somente as regras do ambiente
- Em particular o agente sabe que está em [1,1] e este quadrado é seguro
- A 1ª percepção é [Nada, Nada, Nada, Nada, Nada]
- Então o agente pode concluir que os quadrados adjacentes são seguros

Modelos – Exemplo no Mundo do Wumpus

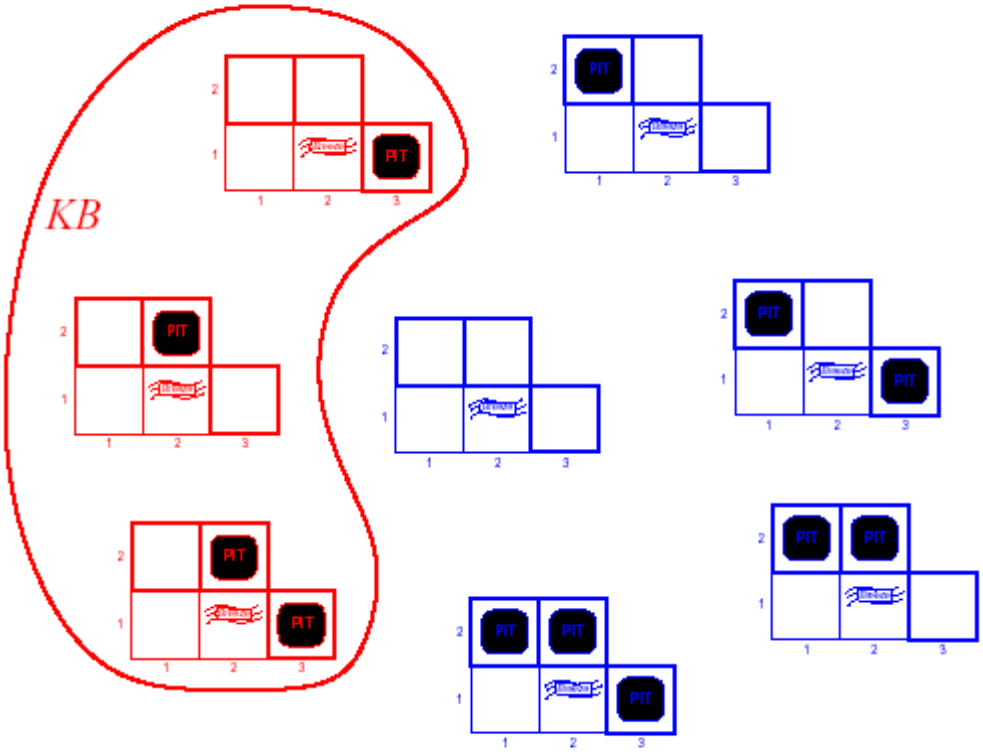
- Situação depois de detectar nada em [1,1], mover-se para direita e detectar brisa em [2,1]
- Considere os modelos possíveis para os ? assumindo somente poços.
  - 3 escolhas booleans → 8 modelos possíveis



Modelos – Exemplo no Mundo do Wumpus

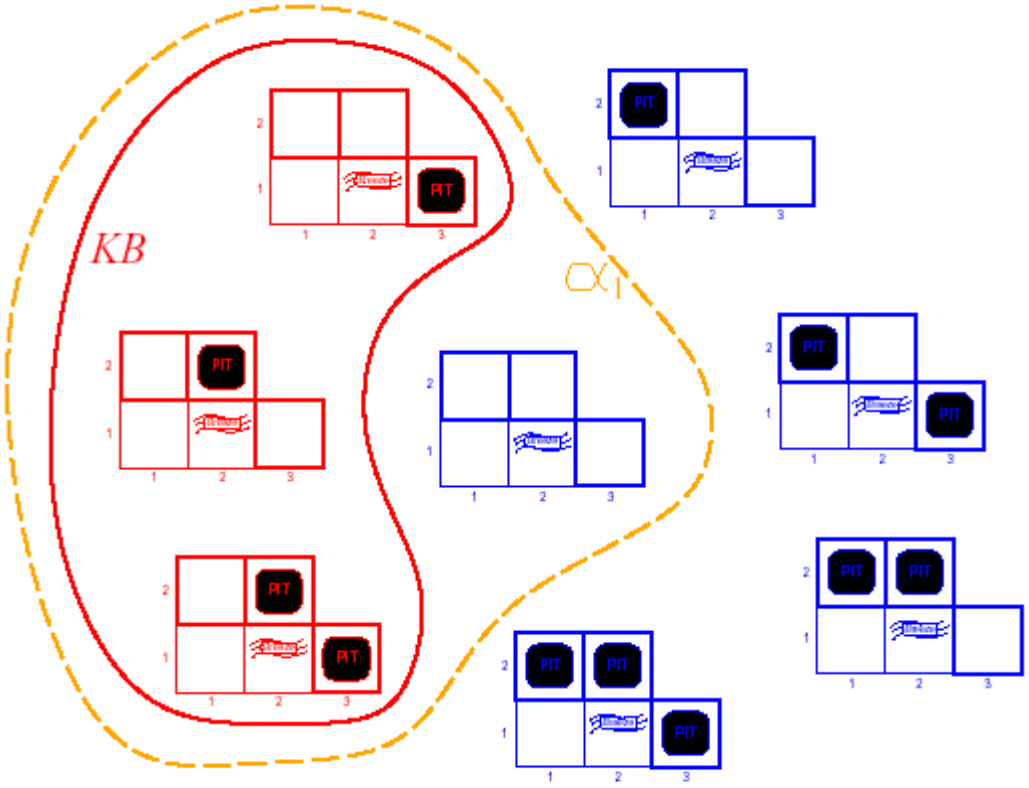


Modelos – Exemplo no Mundo do Wumpus



BC = regras do mundo do *Wumpus* + observações

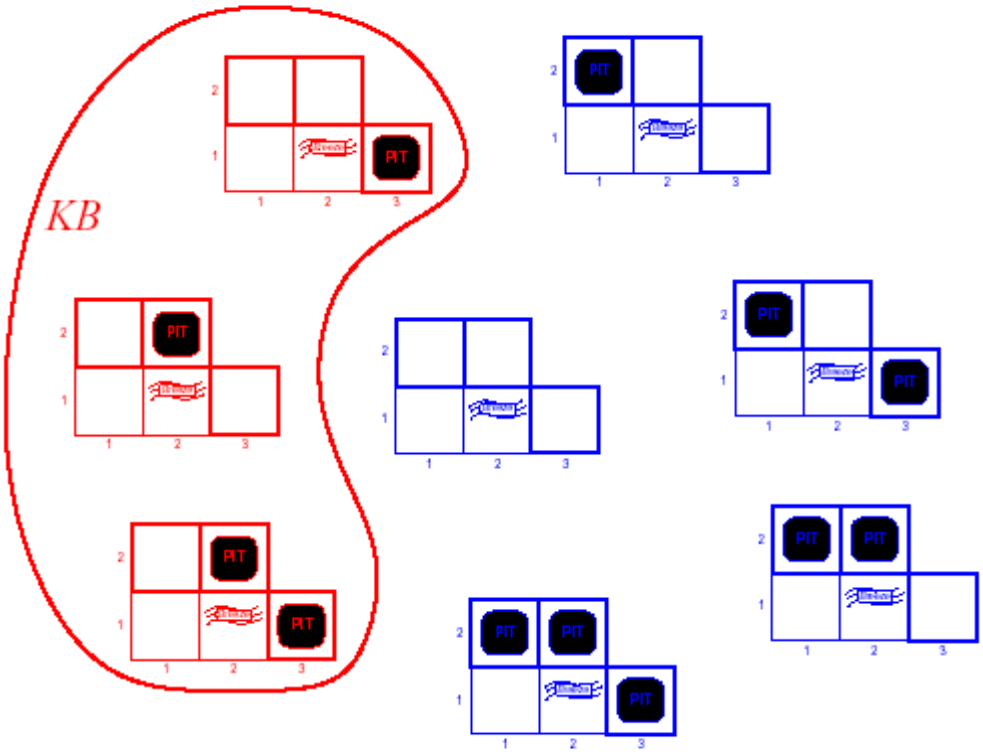
Modelos – Exemplo no Mundo do Wumpus



BC = regras do mundo do *Wumpus* + observações

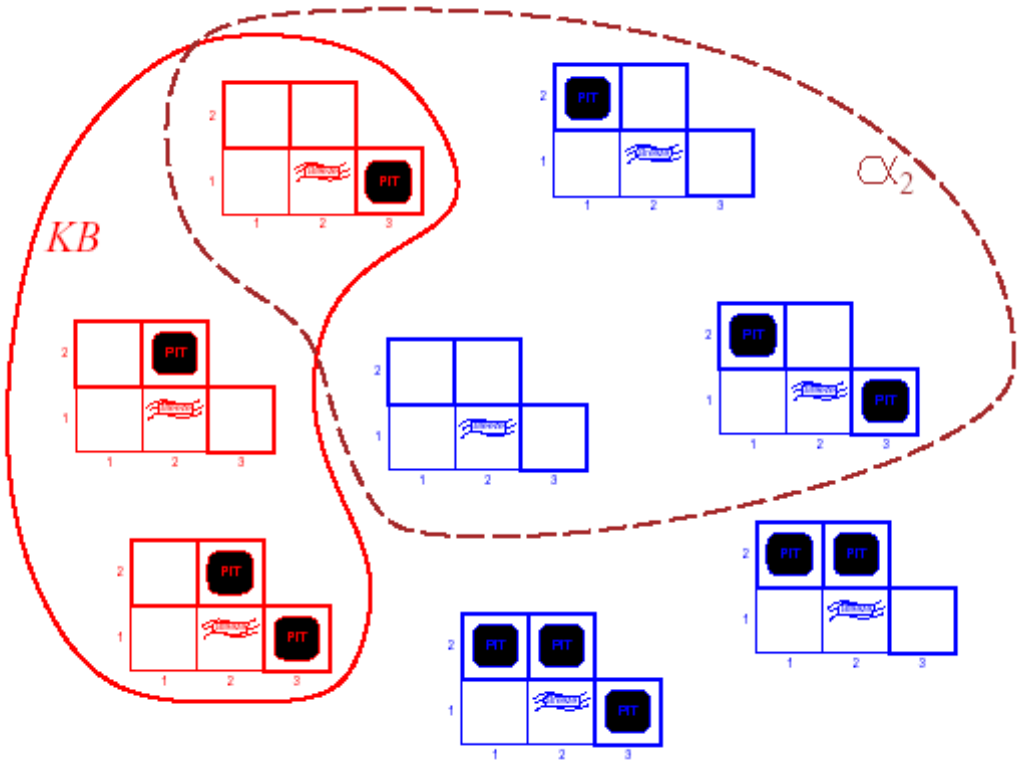
α_1 = "[1,2] é seguro", *BC* |= α_1, verificando os modelos

Modelos – Exemplo no Mundo do Wumpus



BC = regras do mundo do *Wumpus* + observações

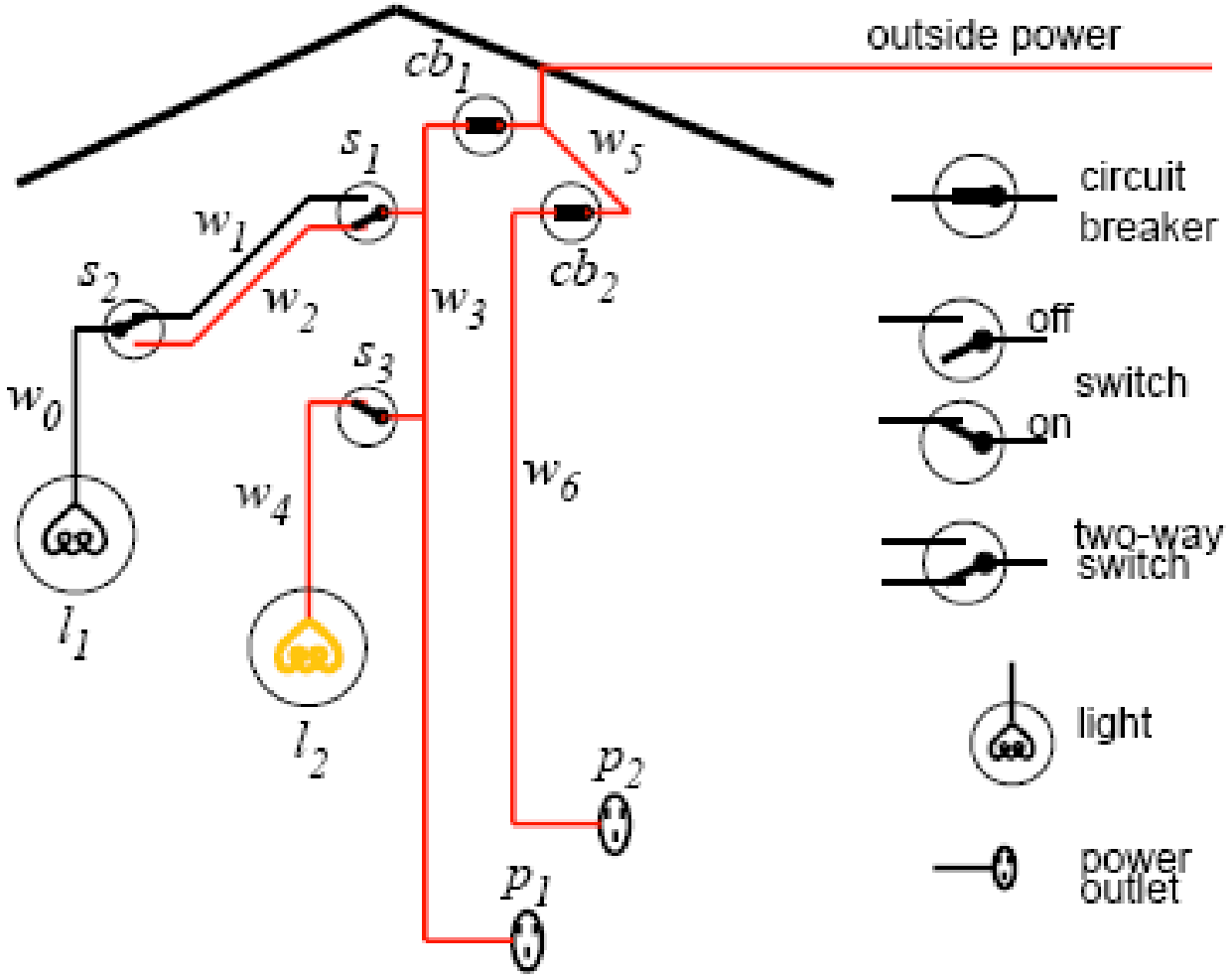
Modelos – Exemplo no Mundo do Wumpus



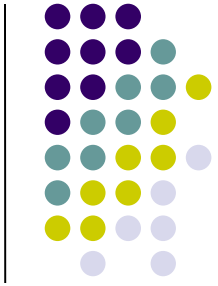
BC = regras do mundo do *Wumpus* + observações

α_2 = "[2,2] é seguro", *BC* $|\neq \alpha_2$

Exemplo axiomatização – Ambiente elétrico



Exemplo axiomatização – Ambiente elétrico



% $light(L)$ is true if L is a light

$light(l_1).$ $light(l_2).$

% $down(S)$ is true if switch S is down

$down(s_1).$ $up(s_2).$ $up(s_3).$

% $ok(D)$ is true if D is not broken

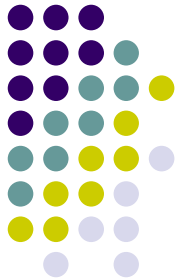
$ok(l_1).$ $ok(l_2).$ $ok(cb_1).$ $ok(cb_2).$

$?light(l_1). \Longrightarrow yes$

$?light(l_6). \Longrightarrow no$

$?up(X). \Longrightarrow up(s_2), up(s_3)$

Exemplo axiomatização – Ambiente elétrico



$connected\_to(X, Y)$ is true if component X is connected to Y

$$connected\_to(w_0, w_1) \leftarrow up(s_2).$$

$$connected\_to(w_0, w_2) \leftarrow down(s_2).$$

$$connected\_to(w_1, w_3) \leftarrow up(s_1).$$

$$connected\_to(w_2, w_3) \leftarrow down(s_1).$$

$$connected\_to(w_4, w_3) \leftarrow up(s_3).$$

$$connected\_to(p_1, w_3).$$

$$?connected\_to(w_0, W). \implies W = w_1$$

$$?connected\_to(w_1, W). \implies no$$

$$?connected\_to(Y, w_3). \implies Y = w_2, Y = w_4, Y = p_1$$

$$?connected\_to(X, W). \implies X = w_0, W = w_1, \ldots$$

Exemplo axiomatização – Ambiente elétrico

% $lit(L)$ is true if the light L is lit

$$lit(L) \leftarrow light(L) \wedge ok(L) \wedge live(L).$$

% $live(C)$ is true if there is power coming into C

$$live(Y) \leftarrow$$
$$connected\_to(Y, Z) \wedge$$
$$live(Z).$$

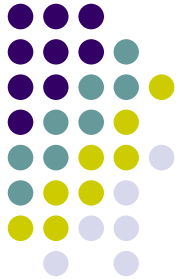
$$live(outside).$$

This is a recursive definition of *live*.

Provas

- Uma **prova** é uma demonstração mecanicamente derivável de que uma fórmula resulta logicamente de uma BC
  - Um **teorema** é uma fórmula que pode ser provada
- Dado um procedimento de prova, *BC* |-- *g* significa que *g* pode ser computado ou derivado de *BC*
  - Relembre-se que *BC* |= *g* significa que *g* é verdadeira em todos os modelos da *BC*
- Um procedimento de prova é correto (***sound***) se tudo que pode ser derivado da BC é uma consequência lógica de BC
  - *BC* |-- *g* implica que *BC* |= *g*
- Um procedimento de prova é **completo** se existe uma prova de cada consequência lógica da BC
  - *BC* |= *g* implica que *BC* |-- *g*

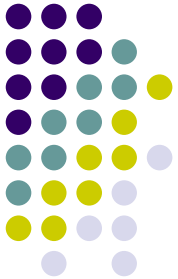
Procedimento de prova fundamental *Bottom-up*



- As derivações são construídas dos fatos que já são conhecidos
- A idéia geral é baseada em uma única **regra de derivação**, uma forma generalizada da regra de inferência chamada de *modus ponens:*
  - Se "$h \leftarrow b_1 \wedge \ldots \wedge b_m$" é uma cláusula na BC, e cada b_i foi derivado, então h pode ser derivado
- Algumas vezes dizemos que estamos fazendo um encadeamento para frente (*forward chaining*) nesta cláusula
  - No sentido de que vamos para frente com o que se sabe até não podemos derivar mais informações

(Esta regra também cobre o caso em que m = 0).

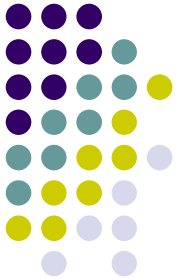
Procedimento de prova fundamental *Bottom-up*



- ***BC* |-- *g* se *g* ∈ *C* ao final deste procedimento:**

C := {} % conjunto de conseqüências
repita
selecione a cláusula "$h \leftarrow b_1 \wedge \ldots \wedge b_m$" na *BC* tal que
$b_i \in C$ para todo *i* e
$h \notin C$;
$C := C \cup \{h\}$
até mais nenhuma cláusula possa ser selecionada.

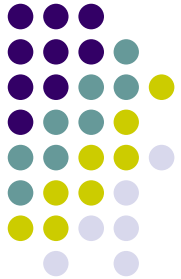
Procedimento de prova fundamental *Bottom-up* - Exemplo



$a \leftarrow b \wedge c.$
$b \leftarrow d \wedge e.$
$b \leftarrow g \wedge e.$
$c \leftarrow e.$
$d.$
$e.$
$f \leftarrow a \wedge g.$

- {}
- {d}
- {d, e}
- {d, e, c}
- {d, e, c, b}
- {d, e, c, b, a}

Procedimento de prova fundamental *Bottom-up* - Exemplo



$a \leftarrow b \wedge c.$

$a \leftarrow e \wedge f.$

$b \leftarrow f \wedge k.$

$c \leftarrow e.$

$d \leftarrow k.$

$e.$

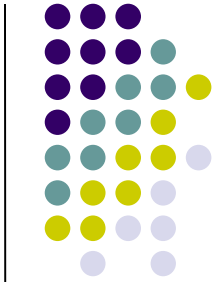
$f \leftarrow j \wedge e.$

$f \leftarrow c.$

$j \leftarrow c.$

- {}
- {e}
- {e, c}
- {e, c, f}
- {e, c, f, j}
- {e, c, f, j, a}

Confiabilidade do Procedimento de Prova *Bottom-up*

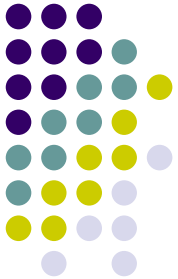


- **Se *BC* |-- *g* então *BC* |= *g***
- Suponha que exista *g* tal que *BC* |-- *g* e *BC* |≠ *g*.
- Seja *h* o primeiro átomo a ser adicionado a *C* que não é verdadeiro em todos os modelos da *BC*. Suponha que *h* não é verdade no modelo *I* da *BC*.
- Tem de existir uma cláusula na *BC* da forma
  - $h \leftarrow b_1 \wedge \ldots \wedge b_m$
- Cada *bi* é verdadeiro em *I*. Portanto, esta cláusula é falsa em *I*. logo, *I* não é um modelo de *BC*.
- Contradição: logo tal *g* não existe.

Ponto Fixo

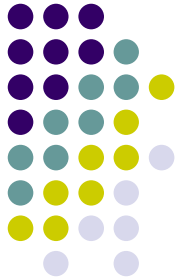
- O *C* gerado ao final do algoritmo de *bottom-up* é chamado de **ponto fixo** (*fixed point*)
  - Qualquer aplicação adicional de uma regra de derivação não muda C
- Seja *I* a interpretação na qual cada elemento do ponto fixo é verdadeiro e cada outro átomo é falso
- *I* é um modelo da *BC*
- Prova: suponha $h \leftarrow b_1 \wedge \ldots \wedge b_m$ na *BC* é falso em *I*. Então *h* é falso e cada b_i é verdadeiro em *I*. Logo, *h* pode ser adicionado a *C*. Contradição com o fato de *C* ser um ponto fixo
- *I* é chamado de Modelo Mínimo

Completude



- ***Se BC |= g então BC |-- g***
- Suponha que *BC |= g*. Então *g* é verdadeiro em todos os modelos da *BC.*
- Logo *g* é verdadeiro no modelo mínimo
- Logo, *g* é gerado pelo algoritmo de *bottom-up*
- Logo, *BC |-- g*

Procedimento de prova fundamental *Top-down*



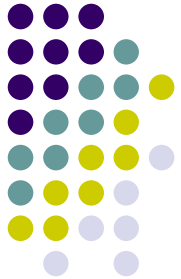
- Idéia: buscar para trás a partir da query para determinar se ela é uma conseqüência lógica da *BC*

- Uma **cláusula resposta** tem a forma:
  - $yes \leftarrow a_1 \wedge a_2 \wedge ... \wedge a_m$

- Uma **resolução SLD** (resolução linear com função de seleção) desta cláusula resposta sobre o átomo a_i com a cláusula:
  - $a_i \leftarrow b_1 \wedge ... \wedge b_p$

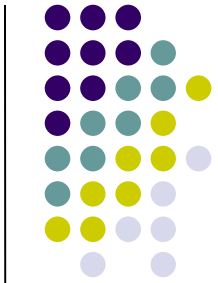
- É a cláusula resposta
  - $yes \leftarrow a_1 \wedge ... \wedge a_{i-1} \wedge b_1 \wedge ... \wedge b_p \wedge a_{i+1} \wedge ... \wedge a_m$.

Derivação



- Uma **resposta** é uma cláusula resposta com m = 0. Isto é, ela é a cláusula resposta *yes* ←.
- Uma **derivação** de uma pergunta "$?q_1 \wedge ... \wedge q_k$" da *BC* é uma seqüência de cláusulas de resposta $\gamma_0, \gamma_1, ..., \gamma_n$ tal que
  - γ_0 é uma cláusula resposta *yes* ← $q_1 \wedge ... \wedge q_k$,
  - γ_i é obtida resolvendo γ_{i-1} com uma cláusula da *BC*, e
  - γ_n é uma resposta.

Procedimento de prova fundamental *Top-down*



Para resolver a pergunta $?q_1 \wedge \ldots \wedge q_k$:

Faça *ac* := "*yes* ← $q_1 \wedge \ldots \wedge q_k$"

repita

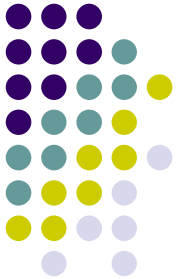
Selecione um a_i do corpo de *ac*;

Escolha a cláusula *C* da *BC* com a_i como cabeça;

Troque a_i no corpo de *ac* pelo corpo de *C*

até *ac* ser uma resposta.

Exemplo de derivação com sucesso



$a \leftarrow b \wedge c.$ $\quad a \leftarrow e \wedge f.$ $\quad b \leftarrow f \wedge k.$

$c \leftarrow e.$ $\quad d \leftarrow k.$ $\quad e.$

$f \leftarrow j \wedge e.$ $\quad f \leftarrow c.$ $\quad j \leftarrow c.$

Query: $?a$

$\gamma_0 : \quad yes \leftarrow a$

$\gamma_1 : \quad yes \leftarrow e \wedge f$

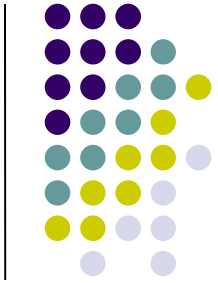
$\gamma_2 : \quad yes \leftarrow f$

$\gamma_3 : \quad yes \leftarrow c$

$\gamma_4 : \quad yes \leftarrow e$

$\gamma_5 : \quad yes \leftarrow$

Exemplo de derivação com falha



$a \leftarrow b \wedge c.$ $\quad a \leftarrow e \wedge f.$ $\quad b \leftarrow f \wedge k.$

$c \leftarrow e.$ $\quad d \leftarrow k.$ $\quad e.$

$f \leftarrow j \wedge e.$ $\quad f \leftarrow c.$ $\quad j \leftarrow c.$

Query: $?a$

$\gamma_0 : \quad yes \leftarrow a$

$\gamma_1 : \quad yes \leftarrow b \wedge c$

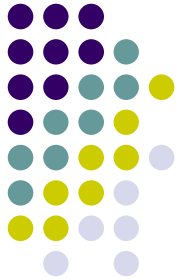
$\gamma_2 : \quad yes \leftarrow f \wedge k \wedge c$

$\gamma_3 : \quad yes \leftarrow c \wedge k \wedge c$

$\gamma_4 : \quad yes \leftarrow e \wedge k \wedge c$

$\gamma_5 : \quad yes \leftarrow k \wedge c$

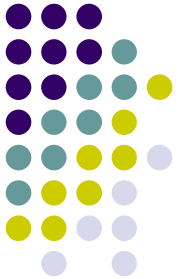
Escolha Não-Determinística



- **Não-determinismo do tipo não importa (*don't-care*)** se uma seleção não leva a uma solução, não há porquê tentar outra alternativa
  - Selecione (no algoritmo *Bottom-up*)
  - A corretude não é afetada pela seleção, mas a eficiência e a terminação são

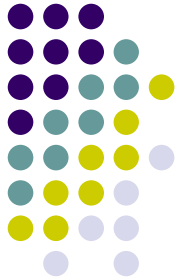
- **Não-determinismo do tipo não sabe (*don't-know*)** se uma escolha não leva a uma solução, outras escolhas podem levar
  - Escolha (no algoritmo *Top-down*)
  - Deve-se buscar por todas as alternativas, pois não sabemos qual delas nos levará à solução

Exercício



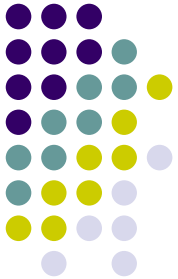
- Considere a seguinte BC
  - $a \leftarrow b \wedge c.$
  - $b \leftarrow d.$
  - $b \leftarrow e.$
  - $c.$
  - $d \leftarrow h.$
  - $e.$
  - $f \leftarrow g \wedge b.$
  - $g \leftarrow c \wedge k.$
  - $j \leftarrow a \wedge b.$
- Determine o ponto fixo desta BC.
- Prove ?*a* usando o procedimento *top-down*.

Raciocinando com variáveis



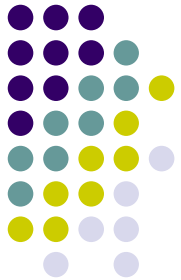
- Uma **instância** de um átomo ou cláusula é obtida pela substituição uniforme de variáveis por termos
  - Todas as instâncias de um variável particular são substituídas pelo mesmo termo
- Uma **substituição** é um conjunto finito da forma $\{V_1/t_1, ..., V_n/t_n\}$, onde cada V_i é uma variável distinta e cada t_i é um termo
  - O elemento V_i/t_i é uma ligação (***binding***) para a variável V_i
- A **aplicação de uma substituição** $\sigma = \{V_1/t_1, ..., V_n/t_n\}$ em um átomo ou cláusula ***e***, escrita como $\boldsymbol{e}\sigma$, é uma instância de ***e*** com cada ocorrência da variável V_i trocada por t_i.

Exemplos de Aplicação



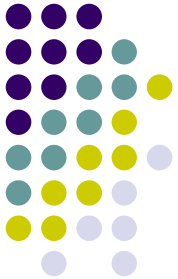
- As seguintes são substituições:
  - $\sigma_1 = \{X/A, Y/b, Z/C, D/e\}$
  - $\sigma_2 = \{A/X, Y/b, C/Z, D/e\}$
  - $\sigma_3 = \{A/V, X/V, Y/b, C/W, Z/W, D/e\}$
- As seguintes mostram algumas aplicações:
  - $p(A, b, C, D)\sigma_1 = p(A, b, C, e)$
  - $p(X, Y, Z, e)\sigma_1 = p(A, b, C, e)$
  - $p(A, b, C, D)\sigma_2 = p(X, b, Z, e)$
  - $p(X, Y, Z, e)\sigma_2 = p(X, b, Z, e)$
  - $p(A, b, C, D)\sigma_3 = p(V, b, W, e)$
  - $p(X, Y, Z, e)\sigma_3 = p(V, b, W, e)$

Unificadores e renomeações



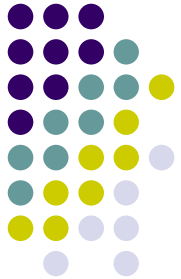
- A substituição σ é um **unificador** de e_1 e e_2 se $e_1\sigma = e_2\sigma$.
  - Exemplo: {X/a, Y/b} é um unificador de t(a, Y, c) e t(X, b, c)
    - t(a, Y, c){X/a, Y/b} = t(a, b, c)
    - t(X, b, c){X/a, Y/b} = t(a, b, c)
- Uma expressão e_1 é uma renomeação de e_2 se elas diferem somente nos nomes das variáveis
  - Neste caso, uma é instância da outra
  - p(X, Y){X/Z, Y/Z} = p(Z, Z)
  - p(X, Y){Z/X, Y/X} = p(X, X)

Exemplos de Unificação



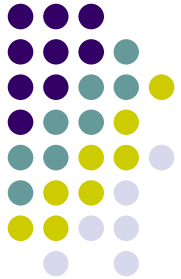
- *p(A,B,C,D)* e *p(X,Y,Z,e) tem como unificadores:*
  - $\sigma_1 = \{X/A, Y/b, Z/C, D/e\}$
  - $\sigma_2 = \{A/X, Y/b, C/Z, D/e\}$
  - $\sigma_3 = \{A/V, X/V, Y/b, C/W, Z/W, D/e\}$
  - $\sigma_4 = \{A/a, X/a, Y/b, C/c, Z/c, D/e\}$
  - $\sigma_5 = \{X/A, Y/b, Z/A, C/A, D/e\}$
  - $\sigma_6 = \{X/A, Y/b, Z/C, D/e, W/a\}$
- As substituições abaixo não são unificadores:
  - $\sigma_7 = \{Y/b, D/e\}$
  - $\sigma_8 = \{X/a, Y/b, Z/c, D/e\}$

Procedimento *Bottom-up*



- Você pode realizar o procedimento *bottom-up* nas instâncias fundamentais das cláusulas
- Confiabilidade (*soundness*) é um corolário direto das confiabilidade fundamental.
- Para completude, construímos um modelo canônico mínimo. Precisamos de uma denotação para constantes.
  - Interpretação de Herbrand: o domínio é o conjunto de constantes (inventamos uma se a BC ou a pergunta não contém uma). Cada constante denota a si mesmo.

Resolução Precisa com Variáveis



- Uma **clausula resposta generalizada** tem a forma

  $yes(t_1,...,t_k) \leftarrow a_1 \wedge a_2 \wedge ... \wedge a_m$,

onde $t_1,...,t_k$ são temos e a_1, ..., a_m são átomos.

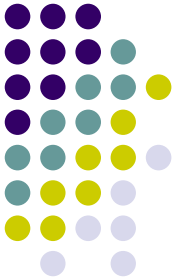
- A **resolução SDL** desta clausula resposta generalizada sobre a_i com a clausula

  $a \leftarrow b_1 \wedge ... \wedge b_p$,

onde a_i e a tem um unificador mais geral θ, é

  $(yes(t_1, ...,t_k) \leftarrow a_1 \wedge...\wedge a_{i-1} \wedge b_1 \wedge...\wedge b_p \wedge a_{i+1} \wedge...\wedge a_m)\theta$.

Para resolver a *query B?* com Variáveis $V_1, \ldots, V_k$:



Faça $ac = yes(V_1,\ldots V_k) \leftarrow B$; // a clausula resposta generalizada

Enquanto *ac* não for uma reposta faça

Suponha que *ac* é $yes(t_1,\ldots t_k) \leftarrow a_1 \wedge a_2 \wedge \ldots \wedge a_m$

Selecione o átomo a_i no corpo de *ac*;

Escolha a clausula $a \leftarrow b_1 \wedge \ldots \wedge b_p$;

Renomeie todas as variáveis em $a \leftarrow b_1 \wedge \ldots \wedge b_p$;

Faça que θ seja o unificador mais geral de a_i e *a*.

Falhe caso não haja unificação;

Faça $ac = (yes(t_1, \ldots,t_k) \leftarrow a_1 \wedge \ldots \wedge a_{i-1} \wedge b_1 \wedge \ldots \wedge b_p \wedge a_{i+1} \wedge \ldots \wedge a_m)\theta$

Fim-enquanto

Exemplo

$live(Y) \leftarrow connected\_to(Y, Z) \wedge live(Z).$ $live(outside)$

$connected\_to(w_6, w_5).$ $connected\_to(w_5, outside).$

$?live(A).$

$yes(A) \leftarrow live(A).$

$yes(A) \leftarrow connected\_to(A, Z_1) \wedge live(Z_1).$

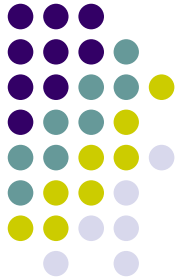
$yes(w_6) \leftarrow live(w_5).$

$yes(w_6) \leftarrow connected\_to(w_5, Z_2) \wedge live(Z_2).$

$yes(w_6) \leftarrow live(outside).$

$yes(w_6) \leftarrow .$

Símbolos de Funções



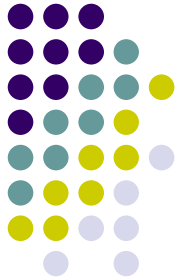
- Quase sempre queremos nos referir a indivíduos em termos de componentes.
  - Exemplo: 4:55 p.m.. Sentenças em Inglês. Uma lista de alunos.

- Assim, estenderemos a noção de termo de tal forma que um termo possa ser $f(t_1,\ldots,t_n)$ onde f é um símbolo de função e os t_i são termos.

- Em uma interpretação com uma atribuição de variáveis, o termo $f(t_1,\ldots,t_n)$ denota um indivíduo do domínio.

- Com um símbolo de função e uma constante podemos referenciar um número infinito de indivíduos.

Listas



- Uma lista é uma seqüência ordenada de elementos.
- Vamos usar a constante *nil* para denotar a lista vazia e a função *cons*(*H*,*T*) para denotar uma lista com o primeiro elemento *H* e o resto-da-lista *T* (esta função não é pré-definida)
- A lista contendo daid, alan e randy é:
  - *cons*(*david*,*cons*(*alan*,(*cons*(*randy*,*nil*)))
- *append*(*X*,*Y*,*Z*) é verdadeiro se a lista *Z* contiver os elementos de *X* seguidos pelos elementos de *Y*
  - *append*(*nil*, *Z*,*Z*).
  - *append*(*cons*(*A*,*X*),*Y*, *cons*(*A*,*Z*)) ← *append*(*X*,*Y*,*Z*)