

## Lógica Proposicional

- **Modelo:** qualquer mundo em que a sentença é verdadeira para alguma interpretação.
- Uma sentença  $\alpha$  é consequência lógica de um KB se os modelos de KB forem todos os modelos de  $\alpha$ .

Modus ponens

$$\frac{\alpha \Rightarrow \beta, \alpha}{\beta}$$

eliminação do E

$$\frac{\alpha_1 \wedge \alpha_2 \wedge \dots \wedge \alpha_n}{\alpha_i}$$

Introdução do E

$$\frac{\alpha_1, \alpha_2, \dots, \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \dots \wedge \alpha_n}$$

Introdução do OU

$$\frac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \dots \vee \alpha_n}$$

- Regras de inferência:

Dupla negação

$$\frac{\neg \neg \alpha}{\alpha}$$

Modus tollens

$$\frac{\alpha \vee \beta, \neg \beta}{\alpha}$$

Resolução

$$\frac{\alpha \vee \beta, \neg \beta \vee \gamma}{\alpha \vee \gamma}$$

## Lógica Proposicional

- **Complexidade**
- Método da tabela verdade é completo: possível enumerar  $2^n$  linhas para a tabela com valores T e F para qualquer prova envolvendo  $n$  símbolos proposicionais.
- Tempo exponencial, mas a maioria dos problemas pode ser resolvido em tempo polinomial usando as regras de inferência.
- Característica de lógica clássica: **monotonicidade**.
- **if**  $KB_1 \models \alpha$  **then**  $(KB_1 \cup KB_2) \models \alpha$
- **Horn sentences** (cláusulas de Horn): possui um procedimento de inferência com tempo polinomial.

## Lógica Proposicional

- Agente para o mundo do wumpus!
- B: brisa, S: mau cheiro, W: wumpus.
- $\neg S_{1,1}, \neg S_{2,1}, S_{1,2}, \neg B_{1,1}, B_{2,1}, \neg B_{1,2}$ : fatos.
- Regras:

$$R_1 : \neg S_{1,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$$

$$R_2 : \neg S_{2,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{2,1} \wedge \neg W_{2,2} \wedge \neg W_{3,1}$$

$$R_3 : \neg S_{1,2} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,2} \wedge \neg W_{1,3}$$

$$R_4 : S_{1,2} \Rightarrow W_{1,1} \vee W_{1,2} \vee W_{1,3} \vee W_{2,2}$$

## Lógica Proposicional

- Como encontrar o wumpus?
- Tabela verdade vai ter 12 símbolos proposicionais:  $S_{1,1}$ ,  $S_{2,1}$ ,  $S_{1,2}$ ,  $W_{1,1}$ ,  $W_{1,2}$ ,  $W_{2,1}$ ,  $W_{2,2}$ ,  $W_{3,1}$ ,  $W_{1,3}$ ,  $B_{1,1}$ ,  $B_{2,1}$ ,  $B_{1,2}$
- $2^{12}$  entradas na tabela!
- Aplicação das regras de inferência.

## Lógica Proposicional

1. Modus Ponens sobre  $\neg S_{1,1}$  ( $R_1$ ):  $\neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$

## Lógica Proposicional

1. Modus Ponens sobre  $\neg S_{1,1}$  ( $R_1$ ):  $\neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$
2. Eliminação E em (1):  $\neg W_{1,1}, \neg W_{1,2}, \neg W_{2,1}$

## Lógica Proposicional

1. Modus Ponens sobre  $\neg S_{1,1}$  ( $R_1$ ):  $\neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$
2. Eliminação E em (1):  $\neg W_{1,1}, \neg W_{1,2}, \neg W_{2,1}$
3. Modus Ponens sobre  $\neg S_{2,1}$  ( $R_2$ ) e elim E:  $\neg W_{1,1}, \neg W_{2,2}, \neg W_{2,1}, \neg W_{3,1}$

## Lógica Proposicional

1. Modus Ponens sobre  $\neg S_{1,1}$  ( $R_1$ ):  $\neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$
2. Eliminação E em (1):  $\neg W_{1,1}, \neg W_{1,2}, \neg W_{2,1}$
3. Modus Ponens sobre  $\neg S_{2,1}$  ( $R_2$ ) e elim E:  $\neg W_{1,1}, \neg W_{2,2}, \neg W_{2,1}, \neg W_{3,1}$
4. Modus Ponens sobre  $S_{1,2}$  ( $R_4$ ):  $W_{1,3} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1}$

## Lógica Proposicional

1. Modus Ponens sobre  $\neg S_{1,1}$  ( $R_1$ ):  $\neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$
2. Eliminação E em (1):  $\neg W_{1,1}, \neg W_{1,2}, \neg W_{2,1}$
3. Modus Ponens sobre  $\neg S_{2,1}$  ( $R_2$ ) e elim E:  $\neg W_{1,1}, \neg W_{2,2}, \neg W_{2,1}, \neg W_{3,1}$
4. Modus Ponens sobre  $S_{1,2}$  ( $R_4$ ):  $W_{1,3} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1}$
5. Resolução unitária com  $\alpha$ :  $W_{1,3} \vee W_{1,2} \vee W_{2,2}$  e  $\beta$ :  $W_{1,1}$ ,  
obtém-se:  $W_{1,3} \vee W_{1,2} \vee W_{2,2}$

## Lógica Proposicional

1. Modus Ponens sobre  $\neg S_{1,1}$  ( $R_1$ ):  $\neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$
2. Eliminação E em (1):  $\neg W_{1,1}, \neg W_{1,2}, \neg W_{2,1}$
3. Modus Ponens sobre  $\neg S_{2,1}$  ( $R_2$ ) e elim E:  $\neg W_{1,1}, \neg W_{2,2}, \neg W_{2,1}, \neg W_{3,1}$
4. Modus Ponens sobre  $S_{1,2}$  ( $R_4$ ):  $W_{1,3} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1}$
5. Resolução unitária com  $\alpha$ :  $W_{1,3} \vee W_{1,2} \vee W_{2,2}$  e  $\beta$ :  $W_{1,1}$ ,  
obtém-se:  $W_{1,3} \vee W_{1,2} \vee W_{2,2}$
6. Res. Unit. com  $\alpha$ :  $W_{1,3} \vee W_{1,2}$  e  $\beta$ :  $W_{2,2}$ . Obtém-se:  
 $W_{1,3} \vee W_{1,2}$

## Lógica Proposicional

1. Modus Ponens sobre  $\neg S_{1,1}$  ( $R_1$ ):  $\neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$
2. Eliminação E em (1):  $\neg W_{1,1}, \neg W_{1,2}, \neg W_{2,1}$
3. Modus Ponens sobre  $\neg S_{2,1}$  ( $R_2$ ) e elim E:  $\neg W_{1,1}, \neg W_{2,2}, \neg W_{2,1}, \neg W_{3,1}$
4. Modus Ponens sobre  $S_{1,2}$  ( $R_4$ ):  $W_{1,3} \vee W_{1,2} \vee W_{2,2} \vee W_{1,1}$
5. Resolução unitária com  $\alpha$ :  $W_{1,3} \vee W_{1,2} \vee W_{2,2}$  e  $\beta$ :  $W_{1,1}$ , obtém-se:  $W_{1,3} \vee W_{1,2} \vee W_{2,2}$
6. Res. Unit. com  $\alpha$ :  $W_{1,3} \vee W_{1,2}$  e  $\beta$ :  $W_{2,2}$ . Obtém-se:  $W_{1,3} \vee W_{1,2}$
7. Res. Unit. com  $\alpha$ :  $W_{1,3}$  e  $\beta$ :  $W_{1,2}$ .
8. Wumpus está na posição  $W_{1,3}$ !

## Lógica Proposicional

- Prova mostrou onde estava o wumpus, mas não mostra **ações**.
- Para as ações:  $A_{1,1} \wedge East_A \wedge W_{1,2} \Rightarrow \neg Forward$ , uma para cada posição e orientação possíveis.
- Lógica proposicional não responde “que ação devo tomar”, mas responde:
  - “Posso mover para a frente?”
  - “Posso virar para a direita?”

## Lógica Proposicional

**function** PROPOSITIONAL-KB-AGENT(percept) **returns** an action

**static:** KB, knowledge base

t, contador, init 0, indica tempo

TELL(KB,MAKE-PERCEPT-SENTENCE(percept,t))

**for each** action in the list of possible actions **do**

**if** ASK(KB,MAKE-ACTION-QUERY(t,action)) **then**

$t \leftarrow t + 1$

**return** action

**end**

## Lógica Proposicional

- Problemas com lógica proposicional:
- muitas proposições para o quadrado 4x4.
- Ex: “não ande para a frente se o wumpus estiver na sua frente” precisa de um conj de 64 regras (16 quadrados x 4 orientações).
- não tem memória do caminho a menos que se represente uma proposição para cada instante no tempo.
- Ex: move para  $A_{2,1}$  se torna verdade e  $A_{1,1}$  se torna falso. Mas pode ser importante guardar o fato de que o agente esteve em  $A_{1,1}$ .
- problema: não sabemos o tempo que vai levar para terminar o jogo.

## Lógica Proposicional

- Exemplo de proposições adicionais:

$$A_{1,1}^0 \wedge East_A^0 \wedge W_{2,1}^0 \Rightarrow \neg Forward^0$$

$$A_{1,1}^1 \wedge East_A^1 \wedge W_{2,1}^1 \Rightarrow \neg Forward^1$$

$$A_{1,1}^2 \wedge East_A^2 \wedge W_{2,1}^2 \Rightarrow \neg Forward^2$$

⋮

- índice no topo de cada símbolo indica tempo.
- para 100 unidades de tempo: 6400 destas regras, somente para dizer: “não mova para a frente se o wumpus estiver lá”.
- lógica de primeira ordem: reduz as 6400 para apenas 1!

## Lógica de Primeira Ordem

- **objetos e relações** entre objetos, **propriedades, funções**.
- **Objetos:** pessoas, casas, números, teorias, Cavaco Silva, cores, jogos de futebol, séculos etc.
- **Relações:** irmão/irmã de, parte de, maior que, tem cor, ocorreu depois, pertence etc.
- **Funções:** pai de, melhor amigo de, vencedor de, um mais que etc.

## Lógica de Primeira Ordem

- Ex: “quadrados vizinhos ao quadrado do wumpus têm mau cheiro”.
  - Objetos: wumpus, quadrado;
  - Propriedade: mau cheiro;
  - Relação: vizinhança.
- Motivação para o uso de lógica de primeira ordem: formalismo mais estudado e melhor entendido que outras abordagens.

## Lógica de Primeira Ordem

- Sintaxe e Semântica.

$S \rightarrow AS \mid SCS \mid QVar, \dots S \mid \neg S \mid (S)$

$AS \rightarrow Pred(Term, \dots) \mid Term = Term$

$Term \rightarrow Func(Term, \dots) \mid Const \mid Var$

$C \rightarrow \Rightarrow \mid \wedge \mid \vee \mid \Leftrightarrow$

$Q \rightarrow \forall \mid \exists$

$Const \rightarrow A \mid X_1 \mid John \dots$

$Var \rightarrow a \mid x \mid s \mid \dots$

$Pred \rightarrow Mother \mid LeftLegOf \mid \dots$

## Lógica de Primeira Ordem

- **Termo:** expressão lógica que se refere a um objeto. Ex: LeftLegOf(ReiJoao).
- **Fórmulas atômicas:** representam fatos. Símbolo de predicado seguido de uma lista de termos entre parênteses. Ex:
  - Irmão(Ricardo, João)
  - Casados(Pai(Ricardo), Mãe(João))
- **Convenção:**  $P(x,y)$ ,  $x$  é  $P$  de  $y$ .
- **Sentenças complexas:**
  - $\text{Irmão}(\text{Ricardo}, \text{João}) \wedge \text{Irmão}(\text{João}, \text{Ricardo})$
  - $\text{MaisVelho}(\text{João}, 30) \vee \text{MaisNovo}(\text{João}, 30)$
  - $\text{MaisVelho}(\text{João}, 30) \Rightarrow \neg \text{MaisNovo}(\text{João}, 30)$

## Lógica de Primeira Ordem

- **Quantificadores:** fazem lógica de primeira ordem ser mais expressiva do que lógica proposicional. Ex:  
 $\forall x \text{ Gato}(x) \Rightarrow \text{Mamifero}(x)$ .
- **Termo ground:** termo sem variáveis.
- quantificador existencial:  $\exists x \text{ Irma}(x, \text{Spot}) \wedge \text{Gato}(x)$ .
- Quantif aninhados: caso mais simples  
 $\forall x, y \text{ Pai}(x, y) \Rightarrow \text{Filho}(y, x)$ , equivalente a  
 $\forall x \forall y \text{ Pai}(x, y) \Rightarrow \text{Filho}(y, x)$ .
- “Todos amam alguém”:  $\forall x \exists y \text{ ama}(x, y)$ .
- “Há alguém que é amado por todos”:  $\exists y \forall x \text{ ama}(x, y)$
- ordem de utilização dos quantif importante.
- dificuldade:  $\forall x [\text{Gato}(x) \vee (\exists x \text{ Irmao}(\text{Ricardo}, x))]$ .

## Lógica de Primeira Ordem

- Relações entre  $\forall$  e  $\exists$ .
- $\forall x \neg Gosta(x, Gatos)$  é equivalente a  $\neg \exists x Gosta(x, Gatos)$ .
- “Todo mundo gosta de sorvete”:  $\forall x Gosta(x, Sorvete)$  ou  $\neg \exists x \neg Gosta(x, Sorvete)$ .
- Leis de De Morgan se aplicam a fórmulas quantificadas e não quantificadas.
- Para efeito de representação não precisamos usar quantificadores ou todos os conectivos.
- **Igualdade:** dois termos referem ao mesmo objeto.

## Extensões e Variações Notacionais

- lógica de mais alta ordem: quantificação feita em cima de funções além de objetos. Ex:
  - $\forall x, y (x = y) \Leftrightarrow (\forall p p(x) \Leftrightarrow p(y))$
  - $\forall f, g (f = g) \Leftrightarrow (\forall x f(x) = g(x))$
- Outras extensões: leitura para casa!
- Variações notacionais: leitura para casa!

## Lógica de Primeira Ordem

- Utilizando lógica de primeira ordem: bancos de dados, prova de teoremas, manipulação de conjuntos.
- Ex: conjuntos.
  - $\forall s \text{ Set}(s) \Leftrightarrow (s = \text{EmptySet}) \vee (\exists x, s_2 \text{ Set}(s_2) \wedge s = \text{AdJoin}(x, s_2))$
  - $\neg \exists x, s \text{ AdJoin}(x, s) = \text{EmptySet}$
  - $\forall x, s \text{ Member}(x, s) \Leftrightarrow s = \text{AdJoin}(x, s)$
  - Notação especial usada para conjuntos:  $[], [x], [x,y], [x,y|l]$ .

## Lógica de Primeira Ordem

- Perguntando e recebendo respostas: TELL e ASK.
- sentenças adicionadas com TELL: assertivas.
- sentenças perguntadas com ASK: consultas ou objetivos.
- respostas podem “instanciar” variáveis: substituições e listas de “bindings”.

## Lógica de Primeira Ordem

- Agente lógico para o mundo do wumpus.
- três tipos de agentes: **reflexos**, **baseados em modelo** e **baseados em objetivos**.
- 1o. passo: definir a interface com o mundo externo
- sentença (interface) típica:  
Percept([Mauchheiro,Brisa,Brilho,N,N],5), onde:
  - arg1: percebe ou não percebe mau cheiro,
  - arg2: percebe ou não percebe brisa,
  - arg3: percebe ou não percebe brilho,
  - arg4: percebe ou não percebe parede,
  - arg5: percebe ou não percebe grito (wumpus sendo morto).
- Ações: Turn(Right), Turn(Left), Forward, Shoot, Grab, Release, Climb.

## Lógica de Primeira Ordem

- Um agente reflexo simples.
- $\forall s, b, u, c, t P([s, b, Brilho, u, c], t) \Rightarrow Action(Grab, t)$
- $\forall b, g, u, c, t P([MauCheiro, b, g, u, c], t) \Rightarrow MauCheiro(t)$
- $\forall s, g, u, c, t P([s, Brisa, g, u, c], t) \Rightarrow Brisa(t)$
- $\forall s, b, u, c, t P([s, b, Brilho, u, c], t) \Rightarrow Ouro(t)$
- $\forall t AtOuro(t) \Rightarrow Action(Grab, t)$

## Lógica de Primeira Ordem

- Limitações de um agente reflexo:
  - não faz parte da percepção deste tipo de agente saber onde está ou se está com o ouro.
  - é incapaz de evitar “loops”. Ex: assuma que o agente conseguiu pegar o ouro e está no caminho de volta para casa. Se passar novamente pelo mesmo quadrado visitado na ida, entra em loop.
  - problema: não está representado neste agente o fato dele estar carregando o ouro e a situação ser diferente da situação da ida.
- precisa de *representação de modificações* no mundo.

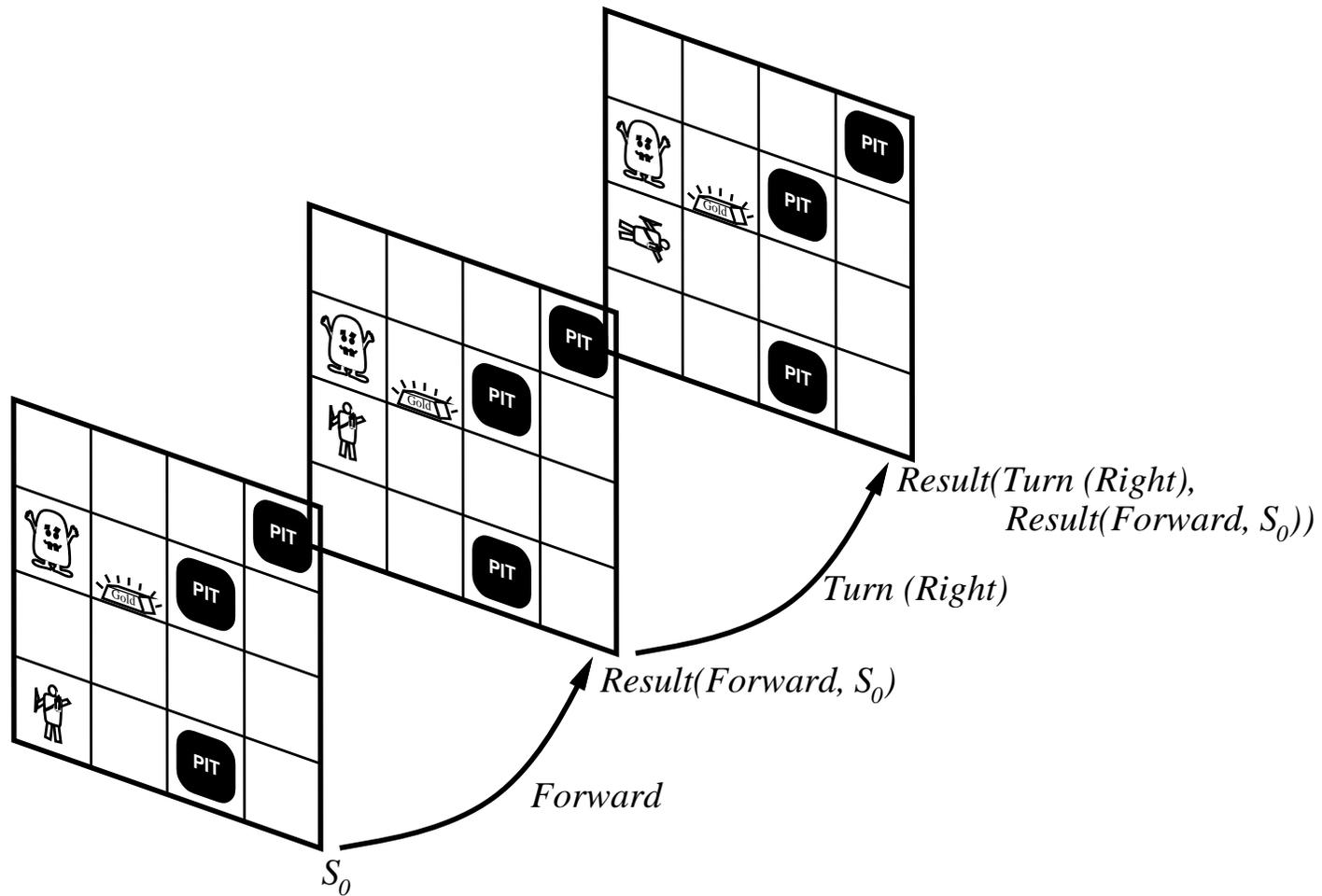
## Lógica de Primeira Ordem

- **Representação de modificações:** uma das áreas mais importantes em representação do conhecimento.
- regras *diacrônicas*.
- representação de *situações* e *ações* não é diferente de representação de objetos e relações.
- *Cálculo de Situações:* forma de descrever modificações em lógica de primeira ordem.

## Lógica de Primeira Ordem

- Considera o mundo como uma sequência de **situações**.
- formato:  $At(\text{Agente}, \text{posição}, \text{situação})$ . Ex:  
 $At(\text{Agent}, [1, 1], S_0) \wedge At(\text{Agent}, [1, 2], S_1)$
- cálculo de situações utiliza  $Result(\text{action}, \text{situation})$  para representar a situação decorrente da execução de uma ação em situação anterior.
- Ex:
  - $Result(\text{Forward}, S_0) = S_1$ ,
  - $Result(\text{Turn}(\text{Right}), S_1) = S_2$ ,
  - $Result(\text{Forward}, S_2) = S_3$

## Lógica de Primeira Ordem



## Lógica de Primeira Ordem

- Ações: são descritas através de seus efeitos: *axiomas de efeito*.
- $Portable(Ouro)$
- $\forall s \text{ AtOuro}(s) \Rightarrow Present(Ouro, s)$
- $\forall x, s \text{ Present}(x, s) \wedge Portable(x) \Rightarrow Holding(x, Result(Grab, s))$
- $\forall x, s \neg Holding(x, Result(Release, s))$
- não suficiente para saber se o agente está segurando o ouro ou continua segurando o ouro.

## Lógica de Primeira Ordem

- necessário: regras para dizer se o mundo continuou o mesmo.
- $\forall a, x, s \text{ Holding}(x, s) \wedge (a \neq \text{Release}) \Rightarrow \text{Holding}(x, \text{Result}(a, s))$
- $\forall a, x, s \neg \text{Holding}(x, s) \wedge (a \neq \text{Grab} \vee \neg(\text{Present}(x, s) \wedge \text{Portable}(x))) \Rightarrow \neg \text{Holding}(x, \text{Result}(a, s))$
- *axiomas de frame.*
- combinação de axiomas de efeito e de frame:  
verdadeiro posteriormente  
 $\Leftrightarrow$  [uma ação fez ser verdadeiro  $\vee$  já era verdadeiro antes]

## Lógica de Primeira Ordem

- $\forall a, s, x \text{ Holding}(x, \text{Result}(a, s)) \Leftrightarrow [(a = \text{Grab} \wedge \text{Present}(x, s) \wedge \text{Portable}(x)) \vee (\text{Holding}(x, s) \wedge a \neq \text{Release})]$
- *axioma do estado sucessor.*
- necessário para cada predicado que pode mudar seu valor no decorrer do tempo.

## Lógica de Primeira Ordem

- Localização do agente.
  - qual é a sua direção: N, S, O, L?
  - Convenção:
    - \* 0 graus anda p/ direita ou esquerda,
    - \* 90 graus anda p/ baixo ou p/ cima.

## Lógica de Primeira Ordem

- Como as posições são mapeadas?
  - $\forall x, y \text{ LocationToward}([x, y], 0) = [x + 1, y]$
  - $\forall x, y \text{ LocationToward}([x, y], 90) = [x, y + 1]$
  - $\forall x, y \text{ LocationToward}([x, y], 180) = [x - 1, y]$
  - $\forall x, y \text{ LocationToward}([x, y], 270) = [x, y - 1]$
  - do mapa é possível dizer se um quadrado está diretamente à frente do agente:
    - \*  $\forall p, l, s \text{ At}(p, l, s) \Rightarrow \text{LocationAhead}(p, s) = \text{LocationToward}(l, \text{Orientation}(p, s))$
    - \* adjacência:
      - $\forall l_1, l_2 \text{ Adj}(l_1, l_2) \Leftrightarrow \exists d \ l_1 = \text{LocationToward}(l_2, d)$

## Lógica de Primeira Ordem

- O que as ações devem fazer com as posições?
  - $\forall a, d, p, s \text{ At}(p, l, \text{Result}(a, s)) = d \Leftrightarrow [(a = \text{Forward} \wedge l = \text{LocationAhead}(p, s) \wedge \neg \text{Wall}(l)) \vee (\text{At}(p, l, s) \wedge a \neq \text{Forward})]$
- O que as ações devem fazer com as orientações?
  - $\forall a, d, p, s \text{ Orientation}(p, \text{Result}(a, s)) = d \Leftrightarrow [(a = \text{Turn}(\text{Right}) \wedge d = \text{Mod}(\text{Orientation}(p, s) - 90, 360)) \vee (a = \text{Turn}(\text{Left}) \wedge d = \text{Mod}(\text{Orientation}(p, s) + 90, 360)) \vee (\text{Orientation}(p, s) = d \wedge \neg(a = \text{Turn}(\text{Right}) \vee a = \text{Turn}(\text{Left})))]$
  - Além disso deve saber o q fazer quando está com o ouro e se o wumpus está vivo ou morto (descrição de *Shoot*).

## Lógica de Primeira Ordem

- Dedução de “propriedades escondidas”.
  - $\forall l, s \text{ At}(\text{Agent}, l, s) \wedge \text{Brisa}(s) \Rightarrow \text{Fresco}(l)$
  - $\forall l, s \text{ At}(\text{Agent}, l, s) \wedge \text{MauCheiro}(s) \Rightarrow \text{MauCheiroso}(l)$
- Regras *sincrônicas* para relacionar propriedades de um estado ao mesmo estado.
  - *Causais* (sistemas baseados em modelos):
    - \*  $\forall l_1, l_2, s \text{ At}(\text{Wumpus}, l_1, s) \wedge \text{Adj}(l_1, l_2) \Rightarrow \text{MauCheiroso}(l_2)$
    - \*  $\forall l_1, l_2, s \text{ At}(\text{Buraco}, l_1, s) \wedge \text{Adj}(l_1, l_2) \Rightarrow \text{Fresco}(l_2)$
  - *Diagnósticas* (sistemas baseados em diagnósticos):
    - \*  $\forall l, s \text{ At}(\text{Agent}, l, s) \wedge \text{Brisa}(s) \Rightarrow \text{Fresco}(l)$
    - \*  $\forall l, s \text{ At}(\text{Agent}, l, s) \wedge \text{MauCheiro}(s) \Rightarrow \text{MauCheiroso}(l)$
    - \*  $\forall l_1, s \text{ MauCheiroso}(l_1) \Rightarrow (\exists l_2 \text{ At}(\text{Wumpus}, l_2, s) \wedge (l_2 = l_1 \vee \text{Adj}(l_1, l_2)))$

## Lógica de Primeira Ordem

- $\forall x, y, g, u, c, s \text{ Percept}([N, N, g, u, c], t) \wedge \text{At}(\text{Agent}, x, s) \wedge \text{Adj}(x, y) \Rightarrow \text{OK}(y)$  (regra fraca).
- $\forall x, t (\neg \text{At}(\text{Wumpus}, x, t) \wedge \neg \text{Buraco}(x)) \Leftrightarrow \text{OK}(x)$  (regra mais forte).
- Conclusão: se os axiomas descrevem *completamente* e *corretamente* a forma como o mundo opera e a forma como as percepções são produzidas, então o procedimento de inferência vai inferir a melhor descrição possível de estados no mundo dadas as percepções.

## Lógica de Primeira Ordem

- Prioridades para as ações: dada uma determinada situação, podemos escolher entre várias ações.
- $\forall a, s \text{ Great}(a, s) \Rightarrow \text{Action}(a, s)$
- $\forall a, s \text{ Good}(a, s) \wedge (\neg \exists b \text{ Great}(b, s)) \Rightarrow \text{Action}(a, s)$
- $\forall a, s \text{ Medium}(a, s) \wedge (\neg \exists b \text{ Great}(b, s) \vee \text{Good}(b, s)) \Rightarrow \text{Action}(a, s)$
- $\forall a, s \text{ Risky}(a, s) \wedge (\neg \exists b \text{ Great}(b, s) \vee \text{Good}(b, s) \vee \text{Medium}(b, s)) \Rightarrow \text{Action}(a, s)$
- sistemas que usam este tipo de regra: *action-value*.

## Lógica de Primeira Ordem

- Descrições anteriores não dizem nada sobre as ações. Apenas classificam as ações.
- ações do tipo Great consistem em pegar o ouro quando este for encontrado e pular para fora da caverna com o ouro.
- ações do tipo Good consistem em mover para uma posição que é OK, mas que não foi ainda visitada.
- ações do tipo Medium consistem em mover para uma posição que está OK, mas que já foi visitada.
- ações do tipo Risky consistem em mover para uma posição em que não se conhece a situação.

## Lógica de Primeira Ordem

- Agente baseado em Objetivos: tenta alcançar os objetivos.
- uma vez que o ouro foi encontrado, a política de movimentação na caverna muda radicalmente: devemos voltar à posição inicial o mais rápido possível.
- $\forall s \text{ Holding}(\text{Ouro}, s) \Rightarrow \text{GoalLocation}([1, 1], s)$
- três formas de encontrar uma sequência de passos que levem a algum objetivo:
  - Inferência
  - Busca (pe, best-first search)
  - Planning: sistemas de raciocínio orientados para ações.