PROGRAMAÇÃO EM LÓGICA NÃO-MONOTÓNICA

Representação de Conhecimento e Raciocínio

- Os agentes lógicos precisam de uma linguagem de Representação de Conhecimento e Raciocínio.
  - Cláusulas de Horn são uma possibilidade.
  - Conhecimento é representado como factos e regras
    - p. p é uma proposição incondicionalmente verdadeira;
    - $q \leftarrow b_1, b_2, \ldots, b_n$. q é uma proposição condicionalmente verdadeira;
      - Uma regra é uma implicação: $b_1 \land b_2 \land \ldots \land b_n \Rightarrow q$
  - Outras alternativas: Lógica de Primeira Ordem (veremos mais à frente), Lógicas de Descrição, etc...

Programação em Lógica (sintaxe)

- Conjuntos de Factos e Regras
  - pessoa(pedro).
  - pessoa(ana).
  - bar(tertúlia).
  - contente(P) ← pessoa(P), bar(B), cerveja(C), frequenta(P,B), vende(B,C), gosta(P,C).
  - tio(X,Y) ← pessoa(X), pessoa(Y), pessoa(Z), filho(Y,Z), irmão(X,Z).
  - inocente(X) ← pessoa(X), ~culpado(X).

Programação em Lógica (sintaxe)

- Um programa em lógica P sobre um conjunto de átomos **A** é um conjunto finito de regras.
- Uma regra (normal), r, é uma expressão da forma

$$a_0 \leftarrow a_1,\ldots, a_m, \sim a_{m+1},\ldots,\sim a_n.$$

- Onde $0 \leq m \leq n$ e cada $a_i \in \mathbf{A}$ é um átomo para $0 \leq i \leq n$.
- Notação:
  - head(r) = a_0
  - body(r) = $\{a_1,\ldots, a_m, \sim a_{m+1},\ldots,\sim a_n\}$
  - body(r)$^+$ = $\{a_1,\ldots, a_m\}$
  - body(r)$^-$ = $\{a_{m+1},\ldots,a_n\}$
  - Por vezes usa-se ":-" em vez de "←", e "not" em vez de "~".
- Um programa é positivo se body(r)$^-$ = { } para todas as regras $r \in P$

Programação em Lógica (semântica)

- Problema: dado um programa em lógica, queremos saber o que é verdadeiro (e o que é falso).
  - Programas sem negação
  - Programas com negação

Programação em Lógica (semântica)

- Programa:

```
pessoa(pedro).
pessoa(ana).
pessoa(rui).
amigo(pedro,ana).
tem_amigos(X) :- pessoa(X), pessoa(Y), amigo(X,Y).*
contente(X) :- pessoa(X), tem_amigos(X).
amigo(X,Y) :- pessoa(X), pessoa(Y), amigo(Y,X).
marciano(X) :- pessoa(X), nasceu_em_marte(X).
```

*regras com variáveis são um shortcut para todas as possíveis instancias

- O que é verdadeiro?

pessoa(pedro)	tem_amigos(pedro)	contente(ana)
pessoa(ana)	contente(pedro)	tem_amigos(rui) ?
pessoa(rui)	amigo(ana,pedro)	marciano(pedro) ?
amigo(pedro,ana)	tem_amigos(ana)	marciano(rui) ?

Programação em Lógica (semântica)

- Um conjunto de átomos X é fechado sobre um programa positivo sse para toda a regra $r \in P$, temos que $head(r) \in X$ sempre que $body(r)^+ \subseteq X$.
  - X corresponde a um modelo de P (visto como uma fórmula de lógica proposicional)
- O modelo mínimo de um programa positivo, denotado por Cn(P), é o menor conjunto de átomos que é fechado sobre um programa P.
  - Cn(P) corresponde ao $\subseteq$-menor modelo de P.
- O conjunto de átomos Cn(P) é o modelo estável de um programa positivo P.

Programação em Lógica (semântica)

- Regras positivas são também conhecidas como cláusulas definidas:
  - Cláusulas de Horn com exatamente um átomo positivo.

$$a_0 \lor \neg a_1 \lor \ldots \lor \neg a_m$$

  - Um conjunto de cláusulas definidas tem um modelo mínimo (único).
- Cláusulas de Horn são cláusulas com no máximo um átomo positivo.
  - Cada cláusula definida é uma clausula de Horn, mas não vice-versa
  - Cláusulas de Horn não definidas podem ser vistas como restrições de integridade
  - Um conjunto de cláusulas de Horn pode ter um modelo mínimo, ou nenhum.
- Este modelo mínimo é a semântica pretendida para um conjunto de tais cláusulas.
  - Dado um programa positivo P, Cn(P) corresponde ao menor modelo do conjunto de cláusulas definidas correspondentes a P.

Programação em Lógica (semântica)

- O modelo mínimo pode ser obtido através de um processo iterativo, partindo do modelo vazio, e acrescentando todas as conclusões imediatas que se podem tirar a partir do que já se concluiu, usando as regras do programa.
- O processo termina quando já não for possível concluir mais nada.

Programação em Lógica (semântica)

Programa:

pessoa(pedro). pessoa(ana). pessoa(rui). amigo(pedro,ana).
tem_amigos(pedro) ← pessoa(pedro), pessoa(ana), amigo(pedro, ana).
tem_amigos(ana) ← pessoa(ana), pessoa(pedro), amigo(ana, pedro).
tem_amigos(rui) ← pessoa(rui), pessoa(pedro), amigo(rui, pedro).
…
contente(pedro) ← pessoa(pedro), tem_amigos(pedro).
contente(ana) ← pessoa(ana), tem_amigos(ana).
contente(rui) ← pessoa(rui), tem_amigos(rui).
amigo(pedro, pedro) ← pessoa(pedro), pessoa(pedro), amigo(pedro, pedro).
amigo(ana, pedro) ← pessoa(ana), pessoa(pedro), amigo(pedro, ana).
amigo(rui, pedro) ← pessoa(rui), pessoa(pedro), amigo(pedro, rui).
…
marciano(pedro) ← pessoa(pedro), nasceu_em_marte(pedro).
marciano(ana) ← pessoa(ana), nasceu_em_marte(ana).
marciano(rui) ← pessoa(rui), nasceu_em_marte(rui).

I0 = {}
I1 = I0 ∪ {pessoa(pedro), pessoa(ana), pessoa(rui), amigo(pedro,ana)}
I2 = I1 ∪ {tem_amigos(pedro), amigo(ana, pedro)}
I3 = I2 ∪ {contente(pedro), tem_amigos(ana)}
I4 = I3 ∪ {contente(ana)}
I5 = I4 ∪ {} = I4

Modelo Mínimo:

M = {pessoa(pedro),pessoa(ana),pessoa(rui),amigo(pedro,ana), amigo(ana,pedro),
tem_amigos(pedro), tem_amigos(ana),contente(pedro), contente(ana)}

Programação em Lógica (semântica)

- Nalguns programas com negação é fácil intuir o único modelo apropriado.
- Programa:

  pessoa(pedro).

  pessoa(rui).

  culpado(pedro).

  inocente(X) ← pessoa(X), ~culpado(X).
- Modelo:

  {pessoa(pedro),pessoa(rui),culpado(pedro),inocente(rui)}

Programação em Lógica (semântica)

- Noutros programas com negação, não parece ser possível intuir um único modelo:
- Programa:

  pessoa(pedro).

  pacífico(X) ← pessoa(X), ~violento(X).

  violento(X) ← pessoa(X), ~pacífico(X).
- Modelo(s):

  ? {pessoa(pedro)} ✗

  ? {pessoa(pedro), violento(pedro), pacífico(pedro)} ✗

  ? {pessoa(pedro), violento(pedro)} ✓

  ? {pessoa(pedro), pacífico(pedro)} ✓

Programação em Lógica (semântica)

- Fórmula lógica:

$$q \wedge (q \wedge \neg r \rightarrow p)$$

Três modelos clássicos:

{p, q}, {q, r}, {p, q, r}

- Programa:

$q \leftarrow$

$p \leftarrow q, \sim r$

Um modelo estável:

{p, q}

- Informalmente, um conjunto de átomos X é um modelo estável de um programa P se:
  - X é um modelo (clássico) de P
  - Todos os átomos em X são justificados por uma regra de P

Programação em Lógica (semântica)

- A redução, P^X, de um programa P relativamente a um conjunto de átomos X é definida por:

$$P^X = \left\{ head(r) \leftarrow body(r)^+ \mid r \in P \wedge body(r)^- \cap X = \varnothing \right\}$$

- Um conjunto de átomos X é um <u>modelo estável de um programa P</u> se

$$Cn\left(P^X\right) = X$$

  - $Cn(P^X)$ é o $\subseteq$-menor modelo de P^X
  - Todo o átomo de X é suportado i.e. é justificado através da aplicação de uma regra de P.

Programação em Lógica (semântica)

- De outra forma, dado um conjunto de átomos X, P^X é obtida a partir de P, eliminando:
  1. Cada regra contendo ~a no seu corpo com $a \in X$, e depois
  2. Todos os átomos negativos da forma ~a dos corpos das restantes regras.
- Esta é a transformação de Gelfond-Lifschitz

Programação em Lógica (exemplo)

- P:

  pessoa(pedro).
  pacífico(pedro) :- pessoa(pedro), not violento(pedro).
  violento(pedro) :- pessoa(pedro), not pacífico(pedro).

- I = {pessoa(pedro), pacífico(pedro)} é Modelo Estável?
- P^I:

  pessoa(pedro).
  pacífico(pedro) :- pessoa(pedro).

- $Cn(P^I)$ = {pessoa(pedro), pacífico(pedro)}

  $Cn(P^I) = I \Rightarrow$ I é Modelo Estável.

Programação em Lógica (exemplo)

- P:

  pessoa(pedro).
  pacífico(pedro) :- pessoa(pedro), not violento(pedro).
  violento(pedro) :- pessoa(pedro), not pacífico(pedro).

- I = {pessoa(pedro), violento(pedro)} é Modelo Estável?
- P^I:

  pessoa(pedro).
  violento(pedro) :- pessoa(pedro).

- $Cn(P^I)$ = {pessoa(pedro), violento(pedro)}

  $Cn(P^I) = I \Rightarrow$ I é Modelo Estável.

Programação em Lógica (exemplo)

- P:

  pessoa(pedro).
  pacífico(pedro) :- pessoa(pedro), not violento(pedro).
  violento(pedro) :- pessoa(pedro), not pacífico(pedro).

- I = {pessoa(pedro)} é Modelo Estável?
- P^I:

  pessoa(pedro).
  pacífico(pedro) :- pessoa(pedro).
  violento(pedro) :- pessoa(pedro).

- $Cn(P^I)$ = {pessoa(pedro),pacífico(pedro),violento(pedro)}
  $Cn(P^I) \neq I \quad \Rightarrow \quad$ I não é Modelo Estável.

Programação em Lógica (exemplo)

- Há regras que têm o efeito de prevenir a existência de alguns modelos estáveis.
- Por exemplo, se tivermos um programa em lógica e quisermos prevenir a existência de modelos estáveis onde, simultaneamente, os átomos a e b sejam verdadeiros e os átomos c e d falsos, podemos acrescentar a seguinte regra, onde α é um átomo novo):

$$\alpha \text{ :- } a, b, \text{not } c, \text{not } d, \text{not } \alpha.$$

Programação em Lógica (exemplo)

- Voltando ao programa anterior:

  pessoa(pedro).

  pacífico(pedro) :- pessoa(pedro), not violento(pedro).

  violento(pedro) :- pessoa(pedro), not pacífico(pedro).
- Se obtivermos o conhecimento que o Pedro não é violento, então temos que eliminar todos os modelos estáveis (mundos) onde o Pedro é violento.
- Isto pode ser obtido adicionando a regra:

  α :- violento(pedro), not α.
- Vamos ver se funciona…

Programação em Lógica (exemplo)

- P:

  pessoa(pedro).
  pacífico(pedro) :- pessoa(pedro), not violento(pedro).
  violento(pedro) :- pessoa(pedro)
  α :- violento(pedro)

- I = {pessoa(pedro), violento(pedro)} é Modelo Estável?
- P^I:

  pessoa(pedro).
  violento(pedro) :- pessoa(pedro).
  α :- violento(pedro).

$Cn(P^I)$ = {pessoa(pedro), violento(pedro), α}

$Cn(P^I) \neq I$ $\Rightarrow$ I não é Modelo Estável.

Programação em Lógica (exemplo)

- P:

  pessoa(pedro).
  pacífico(pedro) :- pessoa(pedro)
  violento(pedro) :- pessoa(pedro), not pacífico(pedro).
  α :- violento(pedro)

- I = {pessoa(pedro), pacífico(pedro)} é Modelo Estável?
- P^I:

  pessoa(pedro).
  pacífico(pedro) :- pessoa(pedro).
  α :- violento(pedro) .

- $Cn(P^I)$ = {pessoa(pedro), pacífico(pedro) }

  $Cn(P^I) = I \quad \Rightarrow \quad$ I é Modelo Estável.

Programação em Lógica (exemplo)

- Se, inversamente, desejamos apenas a existência de modelos estáveis que obedecem a uma determinada condição, usamos uma técnica similar.
- Por exemplo, se temos um programa lógico e queremos apenas permitir a existência de modelos estáveis onde, simultaneamente, os átomos a e b são verdadeiros, e c e d são falsos, podemos adicionar o par de regras (onde α e β são novos átomos):

$$\beta \text{ :- } a, b, \text{not } c, \text{not } d.$$

$$\alpha \text{ :- } \text{not } \beta, \text{not } \alpha.$$

Programação em Lógica (exemplo)

- Voltando ao programa anterior:

  pessoa(pedro).

  pacífico(pedro) :- pessoa(pedro), not violento(pedro).

  violento(pedro) :- pessoa(pedro), not pacífico(pedro).

- Mas, desta vez, aprendemos que Pedro é violento, pelo que devemos manter todos os modelos estáveis (mundos) em que Pedro é violento, eliminando todos os restantes.
- Isso pode ser feito adicionando as regras:

  β :- violento(pedro).

  α :- not β, not α.

- Vamos ver se funciona…

Programação em Lógica (exemplo)

- P:

  pessoa(pedro).
  pacífico(pedro) :- pessoa(pedro), not violento(pedro).
  violento(pedro) :- pessoa(pedro)
  β :- violento(pedro).
  α :- not β, not α.
- I = {pessoa(pedro), violento(pedro), β} é Modelo Estável?
- P^I:

  pessoa(pedro).
  violento(pedro) :- pessoa(pedro).
  β :- violento(pedro) .
- $Cn(P^I)$ = {pessoa(pedro), violento(pedro), β}

  $Cn(P^I) = I \quad \Rightarrow \quad$ I é Modelo Estável.

Programação em Lógica (exemplo)

- P:

  pessoa(pedro).
  pacífico(pedro) :- pessoa(pedro)
  hawk(peter) :- pessoa(pedro), not pacífico(pedro).
  β :- violento(pedro).
  α :-

- I = {pessoa(pedro), pacífico(pedro)} é Modelo Estável?
- P/I: pessoa(pedro).

  pacífico(pedro) :- pessoa(pedro).
  β :- violento(pedro).
  α.

- Cn(P^I) = {pessoa(pedro), pacífico(pedro), α}

  Cn(P^I) $\neq$ I $\Rightarrow$ I não é Modelo Estável.

Programação em Lógica (exemplo)

- Para simplificar a escrita das regras que impedem a existência de modelos estáveis (habitualmente designadas por restrições de integridade), omitem-se todas as utilizações átomo auxiliar α.
- Para impedir a existência de modelos estáveis onde a condição CONDIÇÃO seja verdadeira, acrescentamos a regra:

  :- CONDIÇÃO.

  - Por exemplo, para impedir a existência de modelos estáveis onde, simultaneamente, os átomos a e b sejam verdadeiros, e c e d falsos, acrescentamos a regra:

    :- a, b, not c, not d.

  - Por exemplo, para apenas permitir a existência de modelos estáveis onde, simultaneamente, os átomos a e b sejam verdadeiros, e c e d falsos, acrescentamos o par de regras (onde β é um átomo novo que não aparece em mais lado nenhum):

    β :- a, b, not c, not d.

    :- not β.

Compilação de Condições fechadas

- A fórmula $F_1 \wedge F_2$ é compilada para
  - $p\_F \leftarrow p\_F_1, p\_F_2.$
- A fórmula $F_1 \vee F_2$ é compilada para
  - $p\_F \leftarrow p\_F_1.$
  - $p\_F \leftarrow p\_F_2.$
- A fórmula $\neg F_1$ é compilada para
  - $p\_F \leftarrow \text{not } p\_F_1.$
- O quantificador existential limitado $\exists X{:}domain.F_1$ é compilado para
  - $p\_F \leftarrow domain(X), p\_F_1(X)$
- O quantificador universal limitado $\forall X{:}domain.F_1$
  - $p\_F \leftarrow \text{not } n\_p\_F.$
  - $n\_p\_F \leftarrow domain(X), \text{not } p\_F_1(X).$

Propriedades dos modelos estáveis

- Um programa em lógica pode ter zero, um, ou vários modelos estáveis.
- Se X é um modelo estável de um programa P, então X é um modelo de P (interpretado como uma formula de lógica proposicional).
- Se X e Y são modelos estáveis de um programa (normal), então $X \not\subset Y$.