Programação por Conjuntos de Resposta

Inteligência Artificial

João Leite e Carlos Viegas Damásio

2010/11

Programas em Lógica

- 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(X,Z),
    irmão(Y,Z).
  - inocente(X) :- pessoa(X), not culpado(X).

Semântica

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

Semântica de Programas Positivos

- 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).
```

- 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) ?

Semântica de Programas Positivos

- Modelos mínimos: o modelo (conjunto de átomos verdadeiros) contém o mínimo possível de átomos que se podem concluir a partir do programa.
- Cada programa positivo P tem um e um só modelo mínimo, least(P).
- 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.

Semântica de Programas Positivos

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)}

Semântica de Programas com negação

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

  pessoa(pedro).

  pessoa(rui).

  culpado(pedro).

  inocente(X) :- pessoa(X), not culpado(X).
- Modelo:

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

Semântica de Programas com negação

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

  pessoa(pedro).

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

  violento(X) :- pessoa(X), not pacífico(X).
- Modelo(s):

  ? {pessoa(pedro)} ✗

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

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

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

Semântica de Programas com negação

- Modelos Estáveis
  - Os modelos estáveis são aqueles que são corroborados pelo programa
  - Podem ser vistos como cenários possíveis.
  - Cada programa pode ter zero, um, ou mais modelos estáveis.
- Determinar os Modelos Estáveis
  - É possível verificar se uma interpretação é um modelo estável através da sua comparação com o modelo mínimo de um programa positivo, obtido a partir do programa original através de uma transformação (Transformação de Gelfond-Lifschitz).

Semântica dos Modelos Estáveis

- Transformação de Gelfond-Lifschitz
  - Seja P um programa e I uma interpretação (conjunto de átomos). O programa P/I é obtido a partir de P:
    - eliminando todas as regras de P com “not a” no corpo, tal que $a \in I$;
    - eliminando todos os átomos negativos das restantes regras;
  - O programa P/I é positivo, pelo que tem um único modelo mínimo, least(P/I).

- Modelos Estáveis
  - Seja P um programa e I uma interpretação. I é um modelo estável de P sse:

$$I = least(P/I)$$

Semântica dos Modelos Estáveis

- 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).

- least(P/I) = {pessoa(pedro), pacífico(pedro)}

  least(P/I) = I $\Rightarrow$ I é Modelo Estável.

Semântica dos Modelos Estáveis

- 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).

- least(P/I) = {pessoa(pedro), violento(pedro)}

  least(P/I) = I $\Rightarrow$ I é Modelo Estável.

Semântica dos Modelos Estáveis

- 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).

- least(P/I) = {pessoa(pedro),pacífico(pedro),violento(pedro)}

  least(P/I) $\neq$ I $\Rightarrow$ I não é Modelo Estável.

Modelos Estáveis - Exemplos

- O Programa:

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

  representa uma disjunção, onde uma pessoa (o Pedro) é pacífica ou é violenta. Os dois modelos estáveis são:

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

- Se acrescentássemos o facto pessoa(ana), teríamos 4 modelos estáveis, correspondendo a todas as combinações onde o Pedro e a Ana são pacifistas ou violentos:

  {pessoa(pedro), pessoa(ana), pacífico(pedro), pacifico(ana)}
  {pessoa(pedro), pessoa(ana), violento(pedro), pacifico(ana)}
  {pessoa(pedro), pessoa(ana), pacífico(pedro), violento(ana)}
  {pessoa(pedro), pessoa(ana), violento(pedro), violento(ana)}

Modelos Estáveis - Exemplos

- Se quiséssemos representar uma disjunção entre 3 átomos, tal poderia ser feito da seguinte forma:

```
album(takk).
k7(X) :- album(X), not cd(X), not vinil(X).
cd(X) :- album(X), not k7(X), not vinil(X).
vinil(X) :- album(X), not k7(X), not cd(X).
```

- Este programa representa uma situação onde um álbum tem um suporte (k7, cd ou vinil).
- Tem 3 modelos estáveis:

{album(takk), k7(takk)}
{album(takk), cd(takk)}
{album(takk), vinil(takk)}

Modelos Estáveis - Exemplos

- Também seria possível representar uma situação onde um álbum pode existir em zero, um, dois ou nos três suportes. Para tal, vamos introduzir três predicados auxiliares (n_k7(X), n_cd(X) e n_vinil(X)). O programa é:

```
album(takk).
k7(X) :- album(X), not n_k7(X).
n_k7(X) :- album(X), not k7(X).
cd(X) :- album(X), not n_cd(X).
n_cd(X) :- album(X), not cd(X).
vinil(X) :- album(X), not n_vinil (X).
n_vinil (X) :- album(X), not vinil (X).
```

- Este programa tem oito modelos estáveis. Eles são (omitindo album(takk)):

{k7(takk), cd(takk), vinil(takk)}
{k7(takk), n_cd(takk), vinil(takk)}
{n_k7(takk), n_cd(takk), vinil(takk)}
{k7(takk), n_cd(takk), n_vinil(takk)}
{n_k7(takk), cd(takk), vinil(takk)}
{k7(takk), cd(takk), n_vinil(takk)}
{n_k7(takk), cd(takk), n_vinil(takk)}
{n_k7(takk), n_cd(takk), n_vinil(takk)}

- Se também omitirmos os predicados auxiliares, obtemos:

{k7(takk), cd(takk), vinil(takk)}
{k7(takk), vinil(takk)}
{vinil(takk)}
{k7(takk)}
{cd(takk), vinil(takk)}
{k7(takk), cd(takk)}
{cd(takk)}
{}

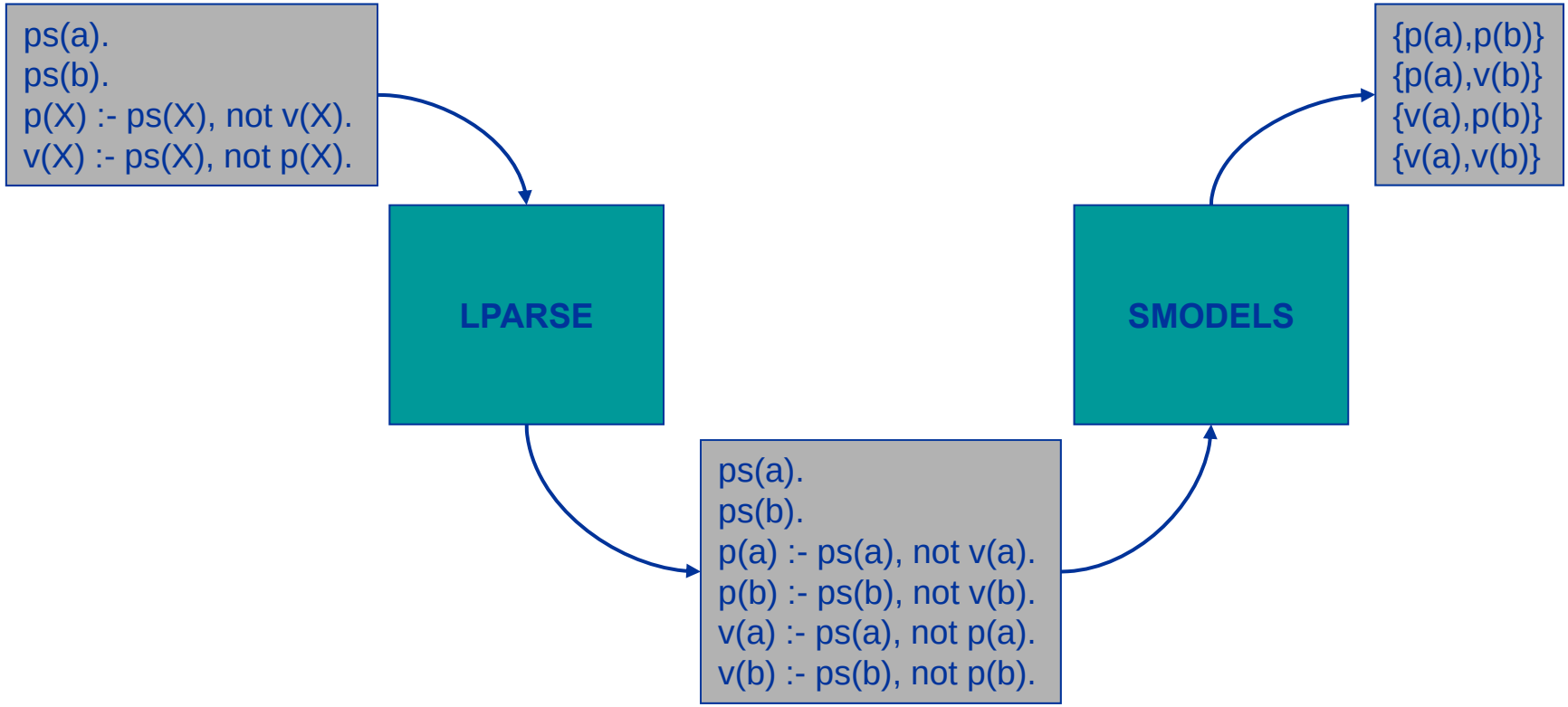
- Tornando mais clara a correspondência entre os modelos estáveis e os cenários possíveis descritos pelo enunciado e programa.

Programação por Conjuntos de Resposta

- A expressividade e carácter declarativo da programação em lógica com a semântica dos modelos estáveis…
- A existência de software que permite, de uma forma eficiente, determinar os modelos estáveis:
  - smodels: http://www.tcs.hut.fi/Software/smodels
  - outros existem...
- …levaram à aplicação deste paradigma na resolução de problemas, levando à introdução da…

Programação por Conjuntos de Resposta

SMODELS



- Linha de comando:

  lparse -nN filename | smodels

- N: número de modelos pretendido. (0 para todos os modelos)

Programação por Conjuntos de Resposta

- Ideia:
  - Representar o problema num programa em lógica de forma a que os seus modelos estáveis (conjuntos de resposta) correspondam às possíveis soluções.

    um modelo estável = uma solução
- 3 passos:
  - determinar o formato das soluções para o problema geral, sendo normal a existência de um (ou mais) predicados que funcionarão como “contentores” da solução;
  - geração de todos os modelos, onde cada modelo representa uma solução hipotética para o problema geral;
  - eliminação dos modelos que não obedecem aos dados do problema concreto.

Modelos Estáveis - Exemplos

- Uma forma alternativa para representar a informação relativa ao suporte dos álbuns é a seguinte:

  album(takk). formato(cd). formato(k7). formato(vinil).

  suporte(A,F) :- album(A), formato(F), not n_suporte(A,F).

  n_suporte(A,F) :- album(A), formato(F), not suporte(A,F).

- Este programa tem os seguintes modelos estáveis (omitindo os átomos album/1, formato/1 e n_suporte/2):

  {suporte(takk,k7), suporte(takk,cd), suporte(takk,vinil)}

  {suporte(takk,cd), suporte (takk,vinil)}

  {suporte(takk,k7), suporte(takk,vinil)}

  {suporte(takk,k7), suporte(takk,cd)}

  {suporte(takk,k7)}

  {suporte(takk,vinil)}

  {suporte(takk,cd)}

  {}

Modelos Estáveis - Exemplos

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

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

- Se, inversamente, apenas quisermos permitir a existência de modelos estáveis que obedeçam a uma dada condição, usa-se uma técnica semelhante.
  - Por exemplo, se tivermos um programa e quisermos apenas permitir a existência de modelos estáveis onde, simultaneamente, os átomos a e b sejam verdadeiros, e c e d falsos, podemos acrescentar o par de regras (onde α e β são átomos novos que não aparecem em mais lado nenhum):

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

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

Modelos Estáveis - Exemplos

- 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 F1 /\ F2 é compilada para
  - p_F :- p_F1, p_F2.
- A fórmula F1 \/ F2 é compilada para
  - p_F :- p_F1.
  - p_F :- p_F2.
- A fórmula ¬F1 é compilada para
  - p_F :- not p_F1.
- O quantificador existential limitado ∃X:domain.F1 é compilado para
  - p_F :- domain(X),p_F1(X)
- O quantificador universal limitado ∀X:domain.F1
  - p_F :- not n_p_F.
  - n_p_F :- domain(X), not p_F1(X).

Modelos Estáveis - Exemplos

- Voltando ao programa anterior:
  album(takk). formato(cd). formato(k7). formato(vinil).
  suporte(A,F) :- album(A), formato(F), not n_suporte(A,F).
  n_suporte(A,F) :- album(A), formato(F), not suporte(A,F).
- Cujos modelos estáveis são (apenas mostrando os átomos suporte/2):
  {suporte(takk,k7), suporte(takk,cd), suporte(takk,vinil)}
  {suporte(takk,cd), suporte (takk,vinil)} {suporte(takk,cd)}
  {suporte(takk,k7), suporte(takk,vinil)} {suporte(takk,vinil)}
  {suporte(takk,k7), suporte(takk,cd)} {suporte(takk,k7)} {}
- Se soubermos que não há nenhum álbum simultaneamente em K7 e CD, podemos acrescentar a seguinte regra (restrição de integridade):
  :- album(A), suporte(A,k7), suporte(A,cd).
- Após a introdução da nova regra, os modelos estáveis passam a ser:
  {suporte(takk,cd), suporte (takk,vinil)} {suporte(takk,cd)} {suporte(takk,k7)}
  {suporte(takk,k7), suporte(takk,vinil)} {suporte(takk,vinil)} {}

Modelos Estáveis - Exemplos

- Continuando com o mesmo programa:
  album(takk). formato(cd). formato(k7). formato(vinil).
  suporte(A,F) :- album(A), formato(F), not n_suporte(A,F).
  n_suporte(A,F) :- album(A), formato(F), not suporte(A,F).
  :- album(A), suporte(A,k7), suporte(A,cd).
- Cujos modelos estáveis são (apenas mostrando os átomos suporte/2):
  {suporte(takk,cd), suporte (takk,vinil)} {suporte(takk,cd)} {suporte(takk,k7)}
  {suporte(takk,k7), suporte(takk,vinil)} {suporte(takk,vinil)} {}
- Se soubermos que cada álbum existe em pelo menos um formato, podemos acrescentar as seguintes regras (restrição de integridade):
  com_formato(A) :- album(A), formato(F), suporte(A,F).
  :- album(A), not com_formato(A).
- Após a introdução das novas regras, os modelos estáveis passam a ser:
  {suporte(takk,cd), suporte (takk,vinil)} {suporte(takk,cd)} {suporte(takk,k7)}
  {suporte(takk,k7), suporte(takk,vinil)} {suporte(takk,vinil)}

Modelos Estáveis - Exemplos

- Continuando com o mesmo programa:
  album(takk). formato(cd). formato(k7). formato(vinil).
  suporte(A,F) :- album(A), formato(F), not n_suporte(A,F).
  n_suporte(A,F) :- album(A), formato(F), not suporte(A,F).
  :- album(A), suporte(A,k7), suporte(A,cd).
  com_formato(A) :- album(A), formato(F), suporte(A,F).
  :- album(A), not com_formato(A).
- Cujos modelos estáveis são (apenas mostrando os átomos suporte/2):
  {suporte(takk,cd), suporte (takk,vinil)} {suporte(takk,cd)} {suporte(takk,k7)}
  {suporte(takk,k7), suporte(takk,vinil)} {suporte(takk,vinil)}
- Se soubermos que o álbum “Takk” existe em vinil, podemos acrescentar o seguinte facto:
  suporte(takk,vinil).
- Após a introdução deste facto, os modelos estáveis passam a ser:
  {suporte(takk,cd), suporte (takk,vinil)} {suporte(takk,k7), suporte(takk,vinil)}
  {suporte(takk,vinil)}

Modelos Estáveis - Exemplos

- Continuando com o mesmo programa, ao qual acrescentamos o novo album “Boy”:
  album(takk). album(boy) formato(cd). formato(k7). formato(vinil).
  suporte(A,F) :- album(A), formato(F), not n_suporte(A,F).
  n_suporte(A,F) :- album(A), formato(F), not suporte(A,F).
  :- album(A), suporte(A,k7), suporte(A,cd).
  com_formato(A) :- album(A), formato(F), suporte(A,F).
  :- album(A), not com_formato(A).
  suporte(takk,vinil).
- Os modelos estáveis são (onde suporte foi substituído por s e omitindo os restantes átomos):
  {s(takk,vinil),s(boy,vinil),s(boy,cd),s(takk,cd)}
  {s(takk,vinil),s(boy,cd),s(takk,cd)} {s(takk,vinil),s(boy,k7)} {s(takk,vinil),s(boy,k7),s(takk,k7)}
  {s(takk,vinil),s(boy,k7),s(takk,cd)} {s(takk,vinil),s(boy,vinil),s(boy,k7),s(takk,cd)}
  {s(takk,vinil),s(boy,vinil),s(takk,cd)} {s(takk,vinil),s(boy,vinil),s(boy,k7)}
  {s(takk,vinil),s(boy,vinil),s(boy,k7),s(takk,k7)} {s(takk,vinil),s(boy,cd),s(takk,k7)}
  {s(takk,vinil),s(boy,cd)} {s(takk,vinil),s(boy,vinil)} {s(takk,vinil),s(boy,vinil),s(boy,cd)}
  {s(takk,vinil),s(boy,vinil),s(boy,cd),s(takk,k7)} {s(takk,vinil),s(boy,vinil),s(takk,k7)}
- Se soubermos que não existem dois álbuns com o mesmo formato, podemos acrescentar a seguinte regra (onde neq(A1, A2) é verdadeiro se A1≠A2):
  :- album(A1), album(A2), formato(F), suporte(A1,F), suporte(A2,F), neq(A1,A2).
- Após a introdução desta regra, os modelos estáveis passam a ser:
  {s(takk,vinil),s(boy,cd)} {s(takk,vinil),s(boy,cd),s(takk,k7)}
  {s(takk,vinil),s(boy,k7),s(takk,cd)} {s(takk,vinil),s(boy,k7)}

Modelos Estáveis - Exemplos

- Continuando com o mesmo programa:

```
album(takk).  album(boy)        formato(cd).        formato(k7).        formato(vinil).
suporte(A,F) :- album(A), formato(F), not n_suporte(A,F).
n_suporte(A,F) :- album(A), formato(F), not suporte(A,F).
:- album(A), suporte(A,k7), suporte(A,cd).
com_formato(A) :- album(A), formato(F), suporte(A,F).
:- album(A), not com_formato(A).
suporte(takk,vinil).
:- album(A1), album(A2), formato(F), suporte(A1,F), suporte(A2,F), neq(A1,A2).
```

- Os modelos estáveis são (apenas mostrando os átomos suporte/2):

  {suporte(takk,vinil), suporte(boy,cd)} {suporte (takk,vinil), suporte(boy,k7)}
  {suporte (takk,vinil), suporte(boy,cd), suporte(takk,k7)}
  {suporte (takk,vinil), suporte(boy,k7), suporte(takk,cd)}

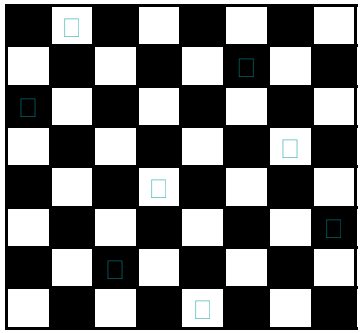
- Se soubermos que existe pelo menos uma k7 e um cd, podemos acrescentar as seguintes regras:

```
ok :- album(A1), album(A2), suporte(A1,cd), suporte(A2,k7).
:- not ok.
```

- Após a introdução desta regra, os modelos estáveis passam a ser (omitindo átomos album/1, formato/1 e n_suporte/2):

  {suporte (takk,vinil), suporte(boy,cd), suporte(takk,k7)}
  {suporte (takk,vinil), suporte(boy,k7), suporte(takk,cd)}

ASP – Rainhas



O problemas das rainhas consiste em colocar 8 rainhas num tabuleiro de xadrez, sem que nenhuma rainha seja atacada por outra i.e. sem que existam duas rainhas na mesma linha, coluna ou diagonal. Solução 1:

coluna(1). … coluna(8). linha(1). … linha(8).

define o domínio de coluna e linha.

in(X,Y) :- coluna(X), linha(Y), not n_in(X,Y).
n_in(X,Y) :- coluna(X), linha(Y), not in(X,Y).

estas duas regras geram todos os modelos resultantes das possíveis combinações onde, para cada célula (X,Y), ou in(X,Y) é verdadeiro, representando que a célula (X,Y) tem uma rainha, ou n_in(X,Y) é verdadeiro, representando que a célula (X,Y) não tem uma rainha.

tem_rainha(X) :- coluna(X), linha(Y), in(X,Y).
:- coluna(X), not tem_rainha(X).

O predicado tem_rainha(X) é definido de tal forma que seja verdadeiro sempre que a coluna X tem pelo menos uma rainha. As duas regras eliminam todos os modelos onde exista uma coluna X sem qualquer rainha.

:- coluna(X), linha(Y), coluna(XX), not eq(X,XX), in(X,Y), in(XX,Y).

esta regra elimina os modelos onde existem rainhas em mais do que uma coluna (X e XX) na mesma linha Y.

:- coluna(X), linha(Y), linha(YY), not eq(Y,YY), in(X,Y), in(X,YY).

esta regra elimina os modelos onde existem rainhas em mais do que uma linha (Y e YY) na mesma coluna X.

:- coluna(X), linha(Y), coluna(XX), linha(YY), not eq(X,XX), not eq(Y,YY),
in(X,Y), in(XX,YY), eq(abs(X-XX),abs(Y-YY)).

esta regra elimina todos os modelos onde existe uma rainha em mais do que uma casa na mesma diagonal.

LPARSE & SMODELS

- n(a;b;c).
  - é equivalente a n(a), n(b), n(c).
- const max = 10
  - define o valor da constante max como sendo igual a 10.
- s(1..max).
  - é equivalente a s(1), s(2), …, s(10).
- eq(X,Y) ⇔ X=Y; lt(X,Y) ⇔ X<Y; ge(X,Y) ⇔ X>=Y; …
- hide p(_,_).
  - esconde os átomos da forma p(_,_) nos modelos gerados pelo smodels;
- hide.
  - esconde todos os átomos nos modelos gerados pelo smodels;
- show q(_,_).
  - invalida a instrução hide para os átomos da forma q(_,_);
- Erros comuns no Lparse, todos eles relacionados com a existência de variáveis, em regras, cujo domínio não está definido de uma forma apropriada:
  - weakely restricted variables;
  - nonrestricted rule;
  - unrestricted variable

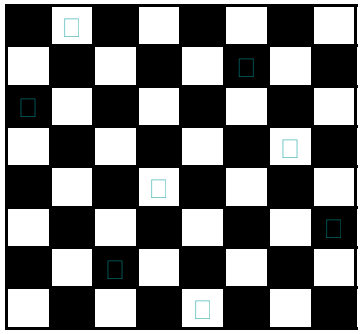
LPARSE & SMODELS

- O Lparse aceita a utilização de sintaxe especial.
- Esta sintaxe permite a especificação da geração selectiva de modelos, eliminando:
  - a necessidade de utilização de alguns átomos auxiliares,
  - de ciclos para gerar modelos,
  - de algumas restrições de integridade.
- A sua utilização permite um grande aumento na eficiência do smodels, devendo ser usada sempre que possível.
- A sintaxe geral é:

$$\mathbf{min\{p(X_1,\ldots,X_n,Y_1,\ldots,Y_m):q(X_1,\ldots,X_n)\}max :- s(Y_1,\ldots,Y_m).}$$

- Leitura: para cada $s(Y_1,\ldots,Y_m)$, gerar todos os modelos possíveis com um mínimo min e um máximo max de átomos $p(X_1,\ldots,X_n,Y_1,\ldots,Y_m)$, onde o domínio de $X_1,\ldots,X_n$ é dado por $q(X_1,\ldots,X_n)$.
- Mais detalhes podem ser encontrados no manual do lparse

ASP – Rainhas



O problemas das rainhas consiste em colocar 8 rainhas num tabuleiro de xadrez, sem que nenhuma rainha seja atacada por outra i.e. sem que existam duas rainhas na mesma linha, coluna ou diagonal. **Solução 2**:

```
coluna(1..8).
```

define o domínio de coluna. Equivalente aos factos coluna(1), coluna(2),…, coluna(8).

```
linha(1..8).
```

define o domínio de linha. Equivalente aos factos linha(1), linha(2),…, linha(8).

```
1{in(X,Y):coluna(X)}1 :- linha(Y).
```

esta regra gera todos os modelos onde cada linha Y está preenchida em exactamente uma coluna X.

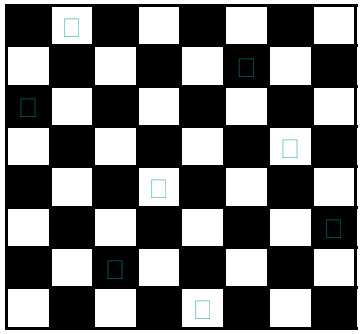
```
:- coluna(X), linha(Y), linha(YY), not eq(Y,YY), in(X,Y), in(X,YY).
```

esta regra elimina todos os modelos onde existe mais do que uma rainha na mesma coluna.

```
:- coluna(X), linha(Y), coluna(XX), linha(YY), not eq(X,XX), not
        eq(Y,YY), in(X,Y), in(XX,YY), eq(abs(X-XX),abs(Y-YY)).
```

esta regra elimina todos os modelos onde existe uma rainha em mais do que uma casa na mesma diagonal.

ASP – Rainhas



O problemas das rainhas consiste em colocar 8 rainhas num tabuleiro de xadrez, sem que nenhuma rainha seja atacada por outra i.e. sem que existam duas rainhas na mesma linha, coluna ou diagonal. **Solução 3**:

```
coluna(1..8).
```

define o domínio de coluna. Equivalente aos factos coluna(1), coluna(2),…, coluna(8).

```
linha(1..8).
```

define o domínio de linha. Equivalente aos factos linha(1), linha(2),…, linha(8).

```
1{in(X,Y):coluna(X)}1 :- linha(Y).
```

esta regra gera todos os modelos onde cada linha Y está preenchida em exactamente uma coluna X.

```
1{in(X,Y):linha(Y)}1 :- coluna(X).
```

esta regra gera todos os modelos onde cada coluna X está preenchida em exactamente uma linha Y.

```
:- coluna(X), linha(Y), coluna(XX), linha(YY), not eq(X,XX), not
        eq(Y,YY), in(X,Y), in(XX,YY), eq(abs(X-XX),abs(Y-YY)).
```

esta regra elimina todos os modelos onde existe uma rainha em mais do que uma casa na mesma diagonal.

ASP – Quadrados Latinos

- O problemas dos quadrados latinos consiste em preencher uma grelha de dimensão n x n com números de 1 a n, de modo a que não existam números repetidos em cada coluna, nem em cada linha.
- Solução 1:

```
const size = 10.
```

  - define a constante size = 10

```
n(1.. size).
```

  - define o domínio de n. Equivalente aos factos n(1),n(2),…,n(size).

```
1{in(X,Y,N):n(N)}1:- n(Y),n(X).
```

  - esta regra gera todos os modelos onde cada célula X,Y está preenchida com exactamente um número N, eliminando os restantes.

```
:- n(X;XX;Y;N), in(X,Y,N), in(XX,Y,N), not eq(X,XX).
```

  - esta regra elimina todos os modelos onde o mesmo número N se encontra em mais do que uma coluna (X e XX) na mesma linha Y.

```
:- n(X;YY;Y;N), in(X,Y,N), in(X,YY,N), not eq(Y,YY).
```

  - esta regra elimina todos os modelos onde o mesmo número N se encontra em mais do que uma linha (Y e YY) na mesma coluna X.

ASP – Quadrados Latinos

- O problemas dos quadrados latinos consiste em preencher uma grelha de dimensão n x n com números de 1 a n, de modo a que não existam números repetidos em cada coluna, nem em cada linha.
- Solução 2 (mais eficiente):

```
const size = 10.
```

  - define a constante size = 10

```
n(1.. size).
```

  - define o domínio de n. Equivalente aos factos n(1),n(2),…,n(size).

```
1{in(X,Y,N):n(N)}1:- n(Y),n(X).
```

  - esta regra gera todos os modelos onde cada célula X,Y está preenchida com exactamente um número N, eliminando os restantes.

```
1{in(X,Y,N):n(Y)}1:- n(X),n(N).
```

  - esta regra gera todos os modelos onde em cada coluna X, cada número N está exactamente numa linha Y, eliminando os restantes.

```
1{in(X,Y,N):n(X)}1:- n(Y),n(N).
```

  - esta regra gera todos os modelos onde em cada linha Y, cada número N está exactamente numa coluna X, eliminando os restantes.

ASP – Sudoku

- O problemas do Sudoku consiste em preencher uma grelha de dimensão 9 x 9 com números de 1 a 9, de modo a que não existam números repetidos em cada coluna, em cada linha, nem em cada uma das 9 sub-grelhas 3 x 3 em que a grelha 9 x 9 se divide.
- O Sudoku é um caso particular do problema dos Quadrados Latinos, com n=9, ao qual se acrescenta a restrição das sub-grelhas de 3 x 3. Assim, podemos partir da solução mais eficiente para os Quadrados Latinos e acrescentar-lhe a nova restrição. Solução:

```
n(1.. 9).
1{in(X,Y,N):n(N)}1:- n(Y),n(X).
1{in(X,Y,N):n(Y)}1:- n(X),n(N).
1{in(X,Y,N):n(X)}1:- n(Y),n(N).
v(1;4;7).
```

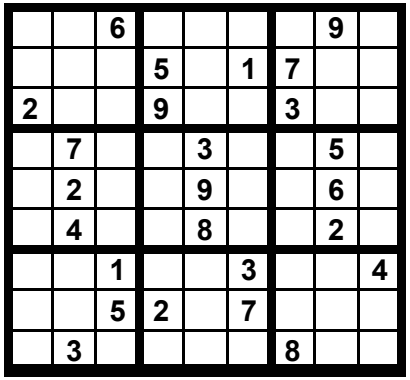
este facto define as coordenadas do canto inferior esquerdo de cada sub-grelha 3 x 3. Equivalente aos factos v(1),v(4), e v(7).

```
:- n(X;Y;X1;Y1;N), v(VX;VY), neq(X,X1), neq(Y,Y1), in(X,Y,N), in(X1,Y1,N),
   X-VX < 3, X1-VX < 3, Y-VY < 3, Y1-VY < 3,
   X >= VX, X1 >= VX, Y >= VY, Y1 >= VY.
```

Esta regra elimina os modelos onde um número N aparece repetido numa sub-grelha 3 x 3. Apenas verifica pares de células onde tanto a linha como a coluna são diferentes. Aqueles onde a linha ou a coluna são iguais já são eliminados pelas regras de geração de modelos anteriores.

ASP – Sudoku

- Normalmente o Sudoku já tem algumas casas preenchidas. Temos portanto que eliminar todos os modelos que não tenham os as casas indicadas preenchidas com os respectivos números.



```
n(1.. 9).
1{in(X,Y,N):n(N)}1:- n(Y),n(X).
1{in(X,Y,N):n(Y)}1:- n(X),n(N).
1{in(X,Y,N):n(X)}1:- n(Y),n(N).
v(1;4;7).
:- n(X;Y;X1;Y1;N), v(VX;VY), neq(X,X1), neq(Y,Y1), in(X,Y,N), in(X1,Y1,N),
    X-VX < 3, X1-VX < 3, Y-VY < 3, Y1-VY < 3,
    X >= VX, X1 >= VX, Y >= VY, Y1 >= VY.
aux :- in(1,7,2),in(2,1,3),in(2,4,4),in(2,5,2),in(2,6,7),in(3,2,5),in(3,3,1),
          in(3,9,6),in(4,2,2),in(4,7,9),in(4,8,5),in(5,4,8),in(5,5,9),in(5,6,3),
          in(6,2,7),in(6,3,3),in(6,8,1),in(7,1,8),in(7,7,3),in(7,8,7),in(8,4,2),
          in(8,5,6),in(8,6,5),in(8,9,9),in(9,3,4).
:- not aux.
```

Modelos Estáveis - Exemplos

- O Programa dos álbuns, usando a sintaxe alternativa do lparse:

  album(takk;boy).

  formato(cd;k7;vinil).

  1{suporte(A,F):formato(F)} :- album(A).

  - esta regra gera todos os modelos onde álbum tem um ou mais formatos (a ausência de um valor para max significa que não há um número máximo de átomos no modelo), eliminando todos os modelos onde um álbum não tenha nenhum formato.

  :- album(A), suporte(A,k7), suporte(A,cd).

  suporte(takk,vinil).

  :- album(A1), album(A2), formato(F), suporte(A1,F), suporte(A2,F), neq(A1,A2).

  ok :- album(A1), album(A2), suporte(A1,cd), suporte(A2,k7).

  :- not ok.

- Os modelos estáveis são (omitindo átomos album/1, formato/1):

  {suporte (takk,vinil), suporte(boy,cd), suporte(takk,k7)}

  {suporte (takk,vinil), suporte(boy,k7), suporte(takk,cd)}

Referências

- Tutoriais:
  - http://costantini.di.univaq.it/wasp.htm
- Manual do Lparse (e Smodels)
  - http://www.tcs.hut.fi/Software/smodels/lparse.ps