PROGRAMAÇÃO POR CONJUNTOS DE RESPOSTA (ANSWER-SET PROGRAMMING)

Programas com Variáveis

- Seja P um programa em lógica
  - Seja **T** o conjunto de termos (s/ variáveis)
  - Seja **A** o conjunto de átomos (s/ variáveis) obtenível a partir de **T**
    - Habitualmente designado por Universo de Herbrand
- Instancias ground de r∈P: conjunto de regras sem variáveis obtidas através da substituição de todas as variáveis de r por elementos de **T**:

$$ground(r) = \left\{ r(\theta) \mid \theta : \mathrm{var}(r) \rightarrow \mathbf{T} \wedge \mathrm{var}(r\theta) = \varnothing \right\}$$

  - var(r) denota o conjunto de todas as variáveis de r;
  - θ é uma substituição (ground)
- Instancias ground de P:

$$ground(P) = \bigcup_{r \in P} ground(r)$$

Programas com Variáveis

- Seja P um programa em lógica com variáveis. Um conjunto de átomos (ground) X é um modelo estável de um programa P se

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

- Uma regra r é segura se cada uma das suas variáveis ocorre num literal de body(r)$^+$.
- Um programa é seguro se todas as suas regras são seguras.
  - Os motores de inferência que calculam modelos estáveis normalmente só aceitam programas seguros.

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

- Para representar uma disjunção entre 3 átomos, poderíamos fazer 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:
  - CLINGO: https://potassco.org/
  - outros existem...
- …levaram à aplicação deste paradigma na resolução de problemas, levando à introdução da…

Programação por Conjuntos de Resposta

(Answer-Set Programming - ASP)

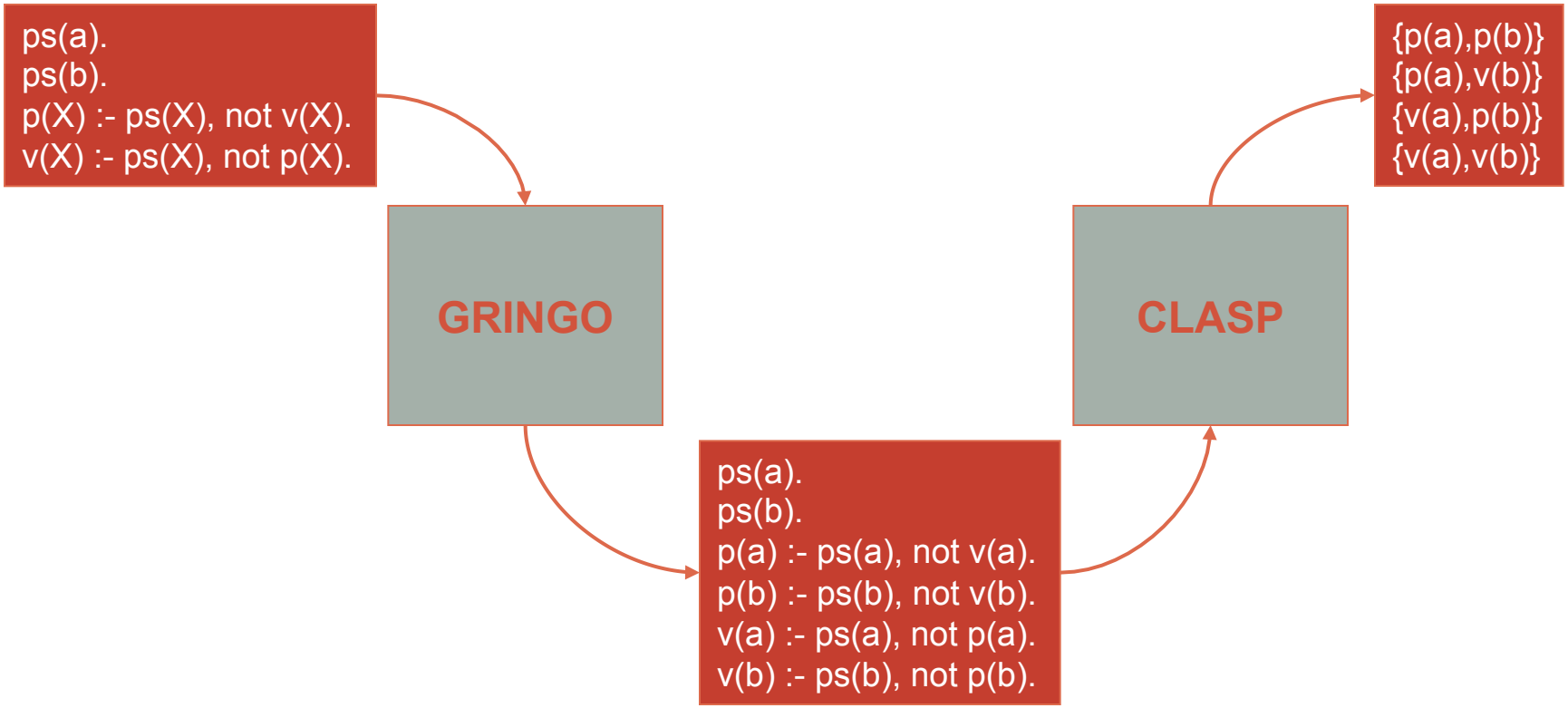
Programação por Conjuntos de Resposta

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

um modelo estável = uma solução

- Programa é composto por:
  - Factos que codificam uma instância
  - Regras que descrevem/codificam a 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 conjuntos de resposta, onde cada representa uma solução hipotética;
  - eliminação dos conjuntos de resposta que não representam soluções do problema;

CLINGO = GRINGO + CLASP



- Linha de comando:
  clingo filename 0

- 0 – devolve todos os conjuntos de resposta.

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)}
{}
```

Programação em Lógica (exemplo)

- Vimos que há regras que têm o efeito de prevenir a existência de alguns modelos estáveis.
- Por exemplo, se 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):

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

- Em geral, impedir a existência de modelos estáveis onde a condição CONDIÇÃO seja verdadeira, acrescentamos a regra:

  :- CONDIÇÃO.

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 a seguinte regra e restrição de integridade:
  ok_s :- suporte(takk,vinil).
  :- not ok_s.
- 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

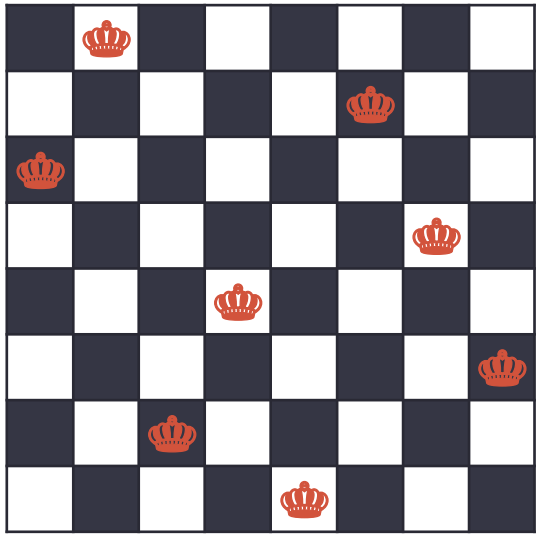
- Continuando com o mesmo programa, ao qual acrescentamos o novo album "Boy":
  album(takk). album(boy) formato(cd). formato(k7). formato(vinil).
  suporte(A,F) :- album(A), formato(F), not n_suporte(A,F).
  n_suporte(A,F) :- album(A), formato(F), not suporte(A,F).
  :- album(A), suporte(A,k7), suporte(A,cd).
  com_formato(A) :- album(A), formato(F), suporte(A,F).
  :- album(A), not com_formato(A).
  ok_s :- suporte(takk,vinil).
  :- not ok_s.
- 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

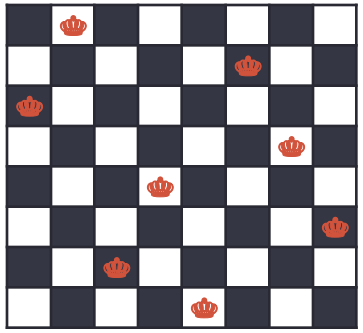
- Continuando com o mesmo programa:
  album(takk). album(boy) formato(cd). formato(k7). formato(vinil).
  suporte(A,F) :- album(A), formato(F), not n_suporte(A,F).
  n_suporte(A,F) :- album(A), formato(F), not suporte(A,F).
  :- album(A), suporte(A,k7), suporte(A,cd).
  com_formato(A) :- album(A), formato(F), suporte(A,F).
  :- album(A), not com_formato(A).
  ok_s :- suporte(takk,vinil).
  :- not ok_s.
  :- 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)}

N-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.



N-Rainhas



- Solução1:

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.

CLINGO = GRINGO + CLASP

- 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 conjuntos de resposta gerados pelo CLINGO;
- hide.
  - esconde todos os átomos nos conjuntos de resposta gerados pelo CLINGO;
- show q(_,_).
  - invalida a instrução hide para os átomos da forma q(_,_);
- Erros comuns, 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

CLINGO = GRINGO + CLASP

- O GRINGO aceita a utilização de sintaxe especial para gerar modelos.

$$\{p_1, p_2, ..., p_n\}.$$

- Leitura: gerar todos os conjuntos de resposta possíveis com subconjuntos de $\{p_1, p_2, ..., p_n\}$.

$$\min\{p_1, p_2, ..., p_n\}\max.$$

- Leitura: gerar todos os conjuntos de resposta possíveis com um mínimo **min** e um máximo **max** de átomos de $\{p_1, p_2, ..., p_n\}$.

$$\{p(X_1,\ldots,X_n):q(X_1,\ldots,X_n)\}.$$

- Leitura: gerar todos os conjuntos de resposta possíveis com instâncias de $p(X_1,\ldots,X_n)$, onde o domínio de $X_1,\ldots,X_n$ é dado por instancias de $q(X_1,\ldots,X_n)$.

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

- Leitura: para cada $s(Y_1,\ldots,Y_m)$, gerar todos os conjuntos de resposta possíveis com instâncias de $p(X_1,\ldots,X_n,Y_1,\ldots,Y_m)$, onde o domínio de $X_1, \ldots,X_n$ é dado por instancias de $q(X_1,\ldots,X_n)$.

CLINGO = GRINGO + CLASP

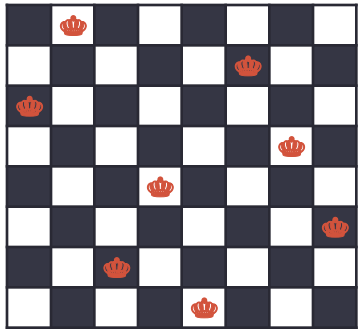
- Caso mais geral:

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

- Leitura: para cada $\mathbf{s(Y_1,\ldots,Y_m)}$, gerar todos os conjuntos de resposta possíveis com um mínimo **min** e um máximo **max** de instâncias de $\mathbf{p(X_1,\ldots,X_n,Y_1,\ldots,Y_m)}$, onde o domínio de $\mathbf{X_1,\ldots,X_n}$ é dado por instancias de $\mathbf{q(X_1,\ldots,X_n)}$, eliminando os restantes.

- Esta sintaxe permite a especificação da geração seletiva 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 CLINGO, devendo ser usada sempre que possível.
- Mais detalhes podem ser encontrados no manual do CLINGO.

N-Rainhas



- Solução2:

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, eliminando os restantes.

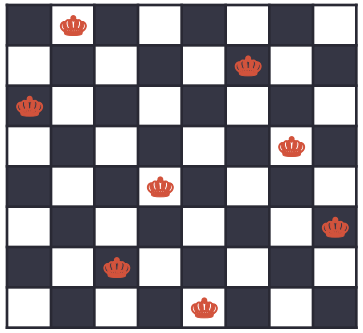
:- coluna(X), linha(Y), linha(YY), not eq(Y,YY), in(X,Y), in(X,YY).

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.

N-Rainhas



- Solução3:

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, eliminando os restantes.

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, eliminando os restantes.

:- 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.

Modelos Estáveis - Exemplos

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

```
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.

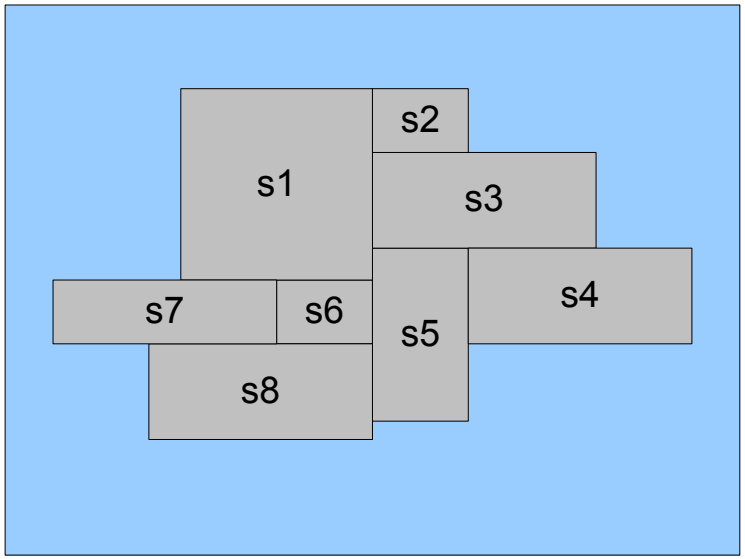
```
:- album(A), suporte(A,k7), suporte(A,cd).
ok_s :- suporte(takk,vinil).
 :- not ok_s.
:- album(A1), album(A2), formato(F), suporte(A1,F), suporte(A2,F), neq(A1,A2).
ok :- album(A1), album(A2), suporte(A1,cd), suporte(A2,k7).
:- not ok.
```

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

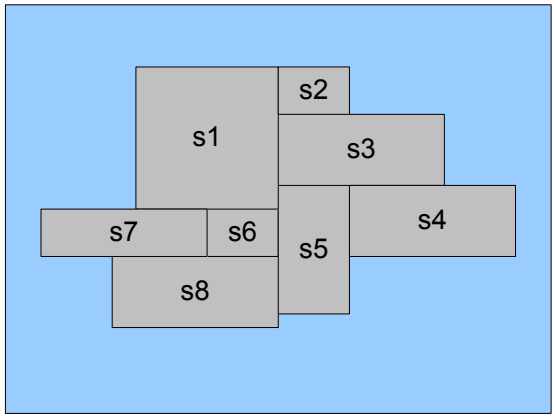
```
{suporte (takk,vinil), suporte(boy,cd), suporte(takk,k7)}
{suporte (takk,vinil), suporte(boy,k7), suporte(takk,cd)}
```

Coloração de Mapas

- Problema: encontrar todas as colorações de um mapa usando no máximo 3 cores, tal que países vizinhos não tenham a mesma cor.



Coloração de Mapas



- Solução1:

state(s1). … state(s8).

border(s1,s2). border(s1,s7). … border(s5,s4).

Estes factos codificam o mapa (grafo) dado.

colour(red). colour(green). colour(blue).

Estes factos codificam as cores disponíveis.

painted(S,C) :- state(S), colour(C), not n_painted(S,C).

n_painted(S,C) :- state(S), colour(C), not painted(S,C).

Estas regras geram todos os conjunto de resposta resultantes das combinações possíveis onde para cada estado (S) e cada cor (C), ou o estado esta pintado com essa cor (painted(S,C) é verdade), ou não está (painted(S,C) é falso, sendo n_painted(S,C) verdadeiro).

is_painted(S) :- state(S), colour(C), painted(S,C).

:- state(S), not is_painted(S).

O predicado is_painted(S) é definido de modo a que pertença a um conjunto de resposta sempre que um estado S tenha pelo menos uma cor. Logo, estas duas regras eliminam todos os conjuntos de resposta onde exista um estado sem qualquer cor.

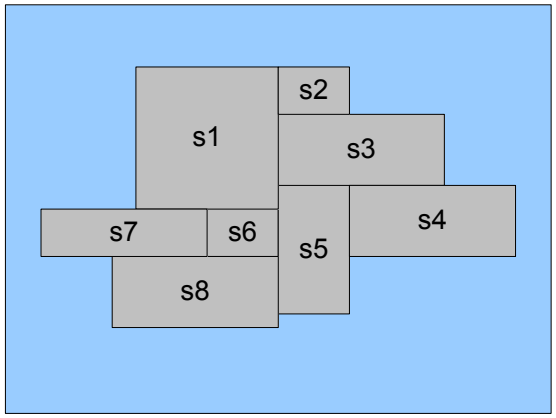
:- state(S), colour(C1), colour(C2), C1!=C2, painted(S,C1), painted(S,C2).

Esta regra elimina todos os conjuntos de resposta onde um estado (S) tem duas cores distintas (C1 e C2).

:- state(S1), state(S2), colour(C), border(S1,S2), painted(S1,C), painted(S2,C).

Esta regra elimina todos os conjuntos de resposta onde dois estados vizinhos (S1 e S2) têm a mesma cor (C).

Coloração de Mapas



- Solução 2:

```
state(s1). … state(s8).
border(s1,s2). border(s1,s7). … border(s5,s4).
```

Estes factos codificam o mapa (grafo) dado.

```
colour(red;green;blue).
```

Estes factos codificam as cores disponíveis.

```
1{painted(S,C):colour(C)}1 :- state(S).
```

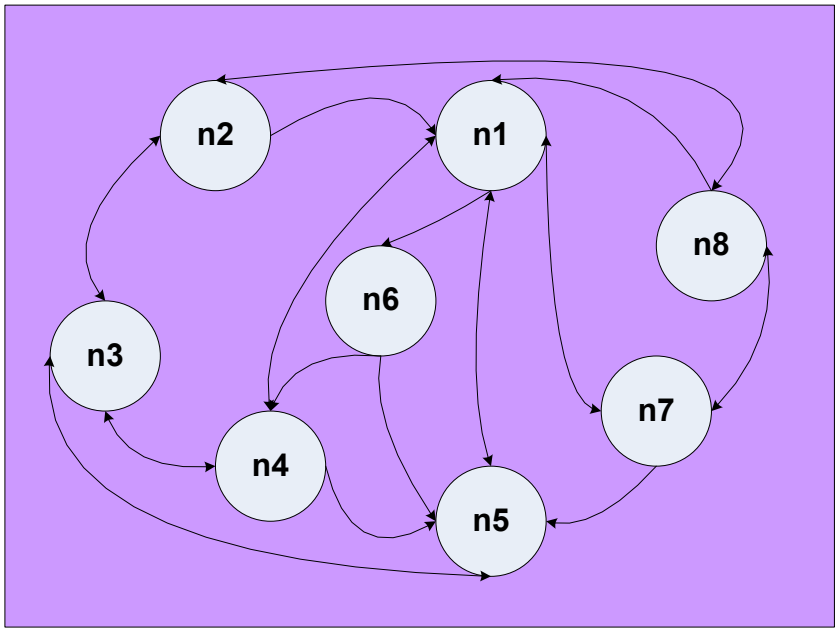
Esta regra gera todos os conjuntos de resposta onde cada estado (S) tem exactamente uma cor (C).

```
:- state(S1), state(S2), colour(C), border(S1,S2), painted(S1,C), painted(S2,C).
```

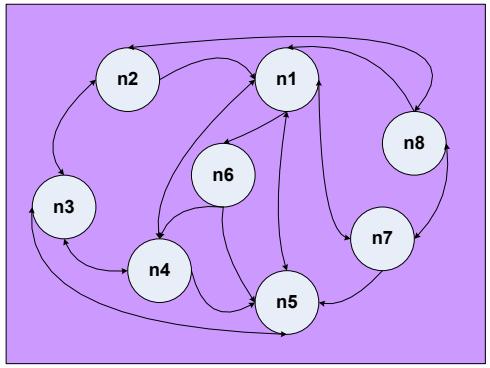
Esta regra elimina todos os conjuntos de resposta onde dois estados vizinhos (S1 e S2) têm a mesma cor (C).

Ciclos Hamiltonianos

- Problema: dado um grafo dirigido, encontrar todos os Ciclos Hamiltonianos i.e. todos os ciclos que tocam cada nó exactamente uma vez.



ASP – Hamiltonian Cycles



node(n1). … node(n8).

edge(n1,n5). edge(n4,n3). … edge(n7,n8).

Estes factos codificam o grafo dado.

1 {in(X,Y) : edge(X,Y)} 1 :- node(X).

Esta regra gera todos os conjuntos de reposta onde o ciclo contém, para cada nó, exactamente um arco a saír.

1 {in(X,Y) : edge(X,Y)} 1 :- node(Y).

Esta regra gera todos os conjuntos de reposta onde o ciclo contém, para cada nó, exactamente um arco a entrar.

reachable(X) :- node(X), in(n1,X).

reachable(Y) :- node(X), node(Y), reachable(X), in(X,Y).

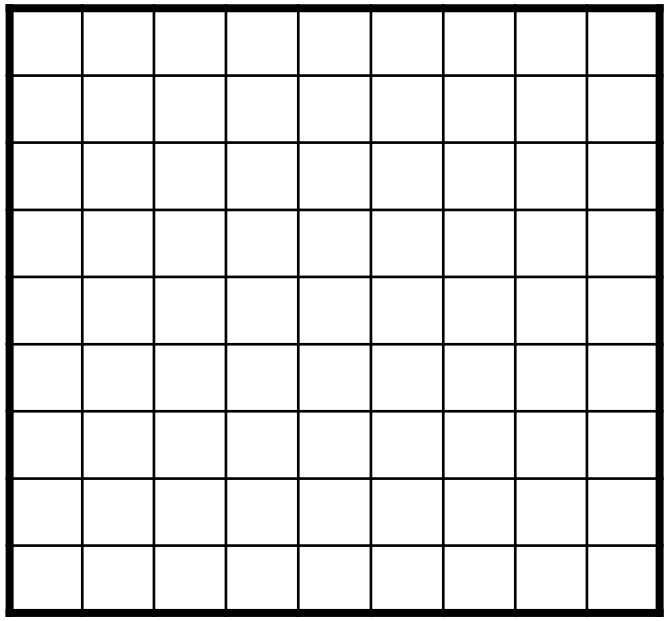
Estas regras definem a noção de acessibilidade a partir de um nó arbitrado (neste caso n1).

:- not reachable(X), node(X).

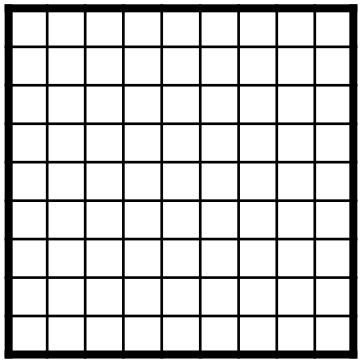
Esta restrição de integridade elimina todos os conjuntos de reposta onde algum nó não é acessível.

Quadrados Latinos

- O problema dos Quadrados Latinos consiste em colocar n símbolos (e.g. números) numa quadrícula n x n de modo a que cada símbolo ocorra precisamente uma vez em cada coluna e em cada linha.



Quadrados Latinos



- 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 conjuntos de resposta onde cada célula (X,Y) esteja preenchida com exactamente um número (N), eliminando todos os outros conjuntos de resposta.

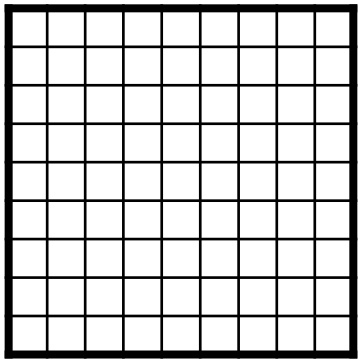
```
:- n(X;XX;Y;N), in(X,Y,N), in(XX,Y,N), X!=XX.
```

  - Esta regra elimina os conjuntos de resposta onde um número (N) é colocado em mais do que uma coluna (X and XX) na mesma linha (Y).

```
:- n(X;YY;Y;N), in(X,Y,N), in(X,YY,N), Y!=YY.
```

  - Esta regra elimina os conjuntos de resposta onde um número (N) é colocado em mais do que uma linha (Y and YY) na mesma coluna (X).

Quadrados Latinos



- Solução 2:

```
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 conjuntos de resposta onde cada célula (X,Y) esteja preenchida com exactamente um número (N), eliminando todos os outros conjuntos de resposta.

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

- Esta regra gera todos os conjuntos de resposta onde em cada coluna (X), cada número (N) é colocado exactamente numa linha (Y), eliminando todos os outros conjuntos de resposta.

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

- Esta regra gera todos os conjuntos de resposta onde em cada linha(Y), cada número (N) é colocado exactamente numa coluna (X), eliminando todos os outros conjuntos de resposta.

Sudoku

- O Sudoku consiste em acabar de preencher com os números de 1 a 9 uma quadrícula 9 x 9 parcialmente preenchida, de modo a que cada símbolo ocorra precisamente uma vez em cada coluna, em cada linha, e em cada uma das 9 caixas 3 x 3 (também designadas por blocos ou regiões).

		6					9	
			5		1	7		
2			9			3		
	7			3			5	
	2			9			6	
	4			8			2	
		1			3			4
		5	2		7			
	3					8		

Sudoku

- O Sudoku é um caso particular dos Quadrados Latinos, com n=9, ao qual acrescentamos a restrição relativa às regiões. Partindo da solução para os Quadrados Latinos, basta acrescentar uma 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).
```

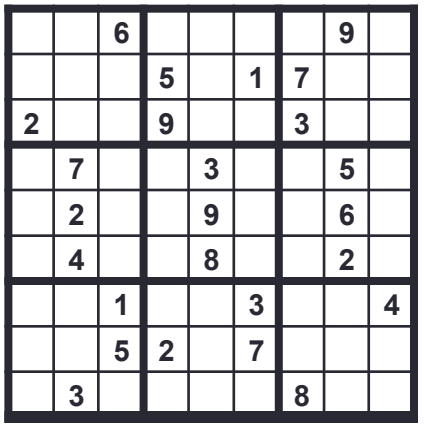
Este facto define as coordenadas da célula inferior esquerda de cada região. Equivalente aos factos v(1),v(4), e v(7).

```
:- n(X;Y;X1;Y1;N), v(VX;VY), X!=X1, 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 restrição de integridade elimina todos os conjuntos de resposta onde um número (N) é colocado em mais do que uma célula na mesma região.

De notar que esta regra apenas verifica pares de células onde tanto as colunas como as linhas das duas células são diferentes, pois os restantes casos já são tratados pelas regras acima.

Sudoku



- Agora basta apenas acrescentar factos correspondentes às células que já estão preenchidas.

```
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), X!=X1, 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.

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).
```