CÓDIGO SAS

Introdução

O SAS é um formato de arquivo usado pelo planejador Fast Downward. Ele resulta da combinação dos arquivos de problema e domínio em um único arquivo que será utilizado pelo algoritmo de busca para encontrar um plano.

Um arquivo SAS possui as seguintes seções:

1. Versão
2. Métrica
3. Variáveis
4. Mutex
5. Estado Inicial
6. Estado Objetivo
7. Operadores
8. Axiomas

Versão

A seção de Versão indica o número da versão usada pela componente de busca para garantir compatibilidade com o tradutor. Neste caso usaremos a versão 3.
Formato geral:

begin_version
<numero_da_versao>
end_version

Métrica

A parte de Métrica indica se o planejador utiliza custos de ação (action costs). Caso a métrica seja definida como 1 (um), são usados custos de ações, e caso seja 0 (zero), não são usados.
Formato geral:

begin_metric
<0 ou 1>
end_metric

Variáveis

Uma variável é composta por grupos de estados mutuamente exclusivos, ou seja, uma variável pode assumir somente um estado de seu grupo por vez.
Forma geral:

<quantidade_de_variaveis>
begin_variable
var0
<nivel_de_axioma>
<quantidade_N_de_estados>
<estado1>
<estado2>
...
<estadoN>
end_variable
...

Variáveis

O estado de uma variável representa um predicado do arquivo de domínio, podendo estar associado a um objeto do arquivo de problema caso seja um predicado com parâmetros.
Forma geral do estado de uma variável:

<estadoX> := Atom <predicado>(<parametros>)
|NegatedAtom <predicado>(<parametros>)

Em que <predicado> é o nome de um predicado e <parametros> é um subconjunto do conjunto de todos os possíveis objetos aceitos como parâmetros desse predicado.

Mutex

A seção de mutex adiciona restrições de exclusão mútua entre váriaveis que não podem ser definidos explicitamente nas seção de variáveis.
Forma geral:

<quantidade_de_grupos_mutex>
begin_mutex_group
<quantidade_N_de_fatos>
<fato1>
<fato2>
...
<fatoN>
end_mutex_group
...

Mutex

Um fato de um mutex corresponde a dois números, o primeiro representando a variável, seguindo a ordem de declaração de variáveis, e o segundo número representado o estado dessa varável, também seguindo a ordem de declaração dos estados. Sendo assim, um fato é da forma:

<fatoX> := <numero_da_variavel> <numero_do_estado_da_variavel>

Mutex

Em caso de não ter mutex. O arquivo sas deverá ter um 0 nesta seção.

Estado Inicial

O estado inicial informa o estado que cada variável declarada assume no início da busca.
Forma geral:

begin_state
<estado_inicial_var0>
<estado_inicial_var1>
...
<estado_inicial_varN-1>
end_state

Estado Objetivo

O estado objetivo informa em qual estado quais variáveis devem assumir no fim da busca.
Forma geral:

begin_goal
<quantidade_de_variaveisM>
<variavelG1> <estado_final_G1>
<variavelG2> <estado_final_G2>
...
<variavelGM> <estado_final_GM>
end_goal

Operadores

Os operadores representam ações que mudam os estados das variáveis, similar às ações do arquivo de domínio. Eles possuem uma lista de condições para que possam ser executados e uma lista de efeitos.
Forma geral:

<quantidade_de_operadores>
begin_operator
<nome_operador0>
<quantidade_de_condicoesN>
<condicao1>
...
<condicaoN>

Operadores

continuação:

<quantidade_de_efeitosM>
<efeito1>
...
<efeitoM>
<custo_do_operador>
end_operator

OBS: caso a seção de métrica seja definida como 0, qualquer valor do campo <custo_do_operador> será tratado como 1.

Operadores

As condições (prevail conditions) de um operador informam quais estados quais variáveis devem assumir para que seja realizado o efeito. A notação das condições é a mesma do par variavel estado descrita no estado objetivo.

<condicaoX> := <variavel> <estado_da_variavel>

Operadores

Cada efeito é uma lista de números em que, o primeiro número denota a quantidade de condições exclusivas daquele efeito. No escopo dessa disciplina. Os efeitos não terão condições próprias, sendo apenas as condições gerais. Com isso, esse primeiro número será 0 para qualquer caso.

Isso é seguido do número que representa a variável afetada pelo efeito. Em seguida vem o estado em que a variável precisa estar para o efeito ser aplicável. Caso o estado da variável não seja relevante, coloca-se -1. Por fim, há o número que representa o novo estado da variável.
De maneira geral:

<efeitoX> := 0 <numero_variavel> <estado_pre-requisito> <novo_estado>

Axiomas

Os axiomas são semelhante em estrutura à seção de operadores. No entanto, essa seção é um pouco mais simples, pois as regras de axioma afetam apenas uma única variável de estado.

Forma geral:

<quantidade_de_axiomas>
begin_rule
<quantidade_de_condicoesN>
<condicao1>
...
<condicaoN>
<efeito1>
end_rule

Axiomas

Em caso de não ter axiomas que são predicados derivados e quantifier operator (forall e exists). O arquivo sas deverá ter um 0 nesta seção.

Exemplo - Domínio

Domain.pddl

```
(define (domain d01)
 (:predicates
  (HelloWorld)
 )
 (:action falar_ola_mundo
  :parameters ()
  :precondition (not(HelloWorld))
  :effect (HelloWorld)
 )
)
```

Exemplo - Problema

Problem.pddl

```
(define (problem p01)
 (:domain d01)
 (:init
  (not(HelloWorld))
 )
 (:goal
  (HelloWorld)
 )
)
```

Exemplo - Variáveis

Domain.pddl

```
(define (domain d01)
 (:predicates
  (HelloWorld)
 )
 (:action falar_ola_mundo
  :parameters ()
  :precondition (not(HelloWorld))
  :effect (HelloWorld)
 )
)
```

Output.sas

```
1
begin_variable
var0
-1
2
Atom helloworld()
NegatedAtom helloworld()
end_variable
```

Exemplo - Estado Inicial

Problem.pddl

```
(define (problem p01)
 (:domain d01)
 (:init
  (not(HelloWorld))
 )
 (:goal
  (HelloWorld)
 )
)
```

Output.sas

```
begin_state
1
end_state

begin_variable
var0
-1
2
0 Atom helloworld()
1 NegatedAtom helloworld()
end_variable
```

Exemplo - Objetivo

Problem.pddl

```
(define (problem p01)
 (:domain d01)
 (:init
  (not(HelloWorld))
 )
 (:goal
  (HelloWorld)
 )
)
```

Output.sas

```
begin_goal
1
0 0
end_goal
```

```
begin_variable
var0
-1
2
0 Atom helloworld()
1 NegatedAtom helloworld()
end_variable
```

Exemplo - Operador

Domain.pddl

```
(define (domain d01)
 (:predicates
  (HelloWorld)
 )
 (:action falar_ola_mundo
  :parameters ()
  :precondition (not(HelloWorld))
  :effect (HelloWorld)
 )
)
```

Output.sas

```
begin_operator
falar_ola_mundo
0
1
0 0 1 0
1
end_operator
```

```
begin_variable
var0
-1
2
0 Atom helloworld()
1 NegatedAtom helloworld()
end_variable
```