PSP
Propositional Satisfiability Problem

Introducción a los algoritmos de planeamiento
2018

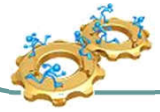
Introducción

- Planning como un Problema de satisfacibilidad booleana (PSP).
- Propositional Satisfiability Problem
  - El problema de determinar si una formula proposicional es satisfacible.

Propositional Satisfiability Problem

- Encontrar una asignación que haga verdadera una formula
- a v b v c

a	b	c	F
T	T	T	T
T	T	F	T
T	F	T	T
T	F	F	T
F	T	T	T
F	T	F	T
F	F	T	T
F	F	F	F



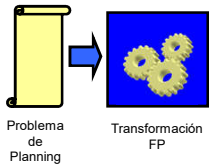
Propositional Satisfiability Problem

- (a v (c b)) b

a	b	c	F
T	T	T	T
T	T	F	T
T	F	T	F
T	F	F	F
F	T	T	T
F	T	F	F
F	F	T	F
F	F	F	F

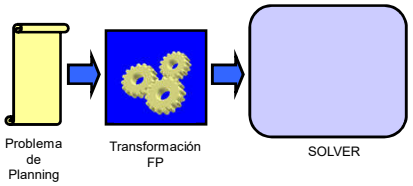
Procedimiento

- El problema de planning es codificado como una formula proposicional



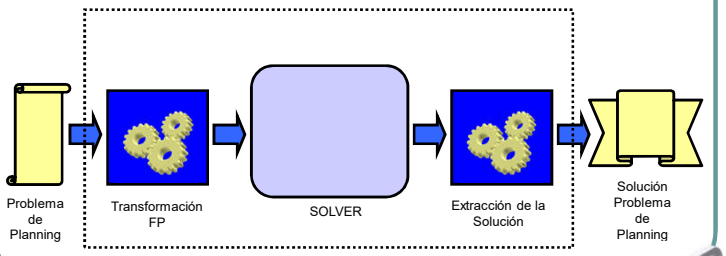
Procedimiento

- Un procedimiento determina si la formula es satifacible asignando valores a las variables proposicionales



Procedimiento

- Un plan es *extraído* de las asignaciones realizadas por el procedimiento



Definiciones

- Problema de planning:

  $P = (\Sigma, s_0, S_g)$ donde

  $\Sigma = (S, A, \gamma)$ Dominio de planning

  S = Conjunto de Estados

  A = Conjunto de Acciones

  γ = Función de Transición deterministica

  s_0 = Estado Inicial

  S_g = Estado final

Definiciones

- Formulas Proposicionales

  Modelo : Asignación de valores de verdad a las variables de las formulas, de manera tal que la formula es TRUE

  Satisfiability Problem : problema de determinar si una formula tiene un *modelo*

Estados como formulas Proposicionales

- Se utilizan formulas proposicionales que representan los hechos que se posee en los estados

  Por ejemplo

  at(r1, l1) ∧ ¬cargado(r1)

  "El robot r1 esta en el lugar l1 y no esta cargado"

  Un modelo seria

  M ={ at(r1, l1) : true, cargado(r1) : false }

Problemas

Que sucede con la formula si tenemos dos lugares (l1 y l2) ?
Aparece **at(r1,l2)**

La formula posee 2 Modelos

M1 ={ at(r1, l1) : true, cargado(r1) : false , at(r1, l2) : true }
M2 ={ at(r1, l1) : true, cargado(r1) : false , at(r1, l2) : false }

Intuitivamente M2 es el modelo que tenemos en mente cuando se escribió la formula puesto el robot no puede estar en dos lugares al mismo tiempo

Descripción completa del estado

at(r1, l1) ∧ ¬ at(r1, l2) ∧ ¬cargado(r1)

Problemas

Una formula puede representar conjuntos de estados

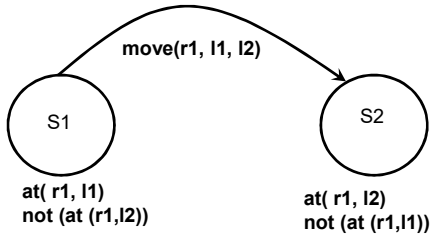
[(at(r1, l1) ∧ ¬ at(r1, l2))
(¬ at(r1, l1) ∧ at(r1, l2))]
∧¬cargado(r1)

Una formula representa a un estado (o varios) pero no codifica la dinámica del sistema

Transiciones como formulas Proposicionales

- Una acción se ve como una función

  S x A --> S



Problemas del modelamiento

at(r1, l1) not (at (r1,l2))

at(r1, l2) not (at (r1,l1))

¿Cómo modelar los distintos Estados?

Representar el estado

Se busca modelar que un hecho que se mantiene en un estado no se mantiene en otro

Se incorpora una variable que represente al estado

at(r1, l1,s1) not (at (r1,l2,s1)) at(r1, l2,s2) not (at (r1,l1,s2))

Modelar las acciones

at(r1, l1,s1) not (at (r1,l2,s1)) at(r1, l1,s2) not (at (r1,l2,s2))

Modela la transición.

Falta modelar que acción produjo la transición.

move(r1, l1, l2, s1) at(r1, l1,s1) not (at (r1,l2,s1))
at(r1, l2,s2) not (at (r1,l1,s2))

Siempre se requiere identificar CUANDO es ejecutada la acción

El problema de planning como formulas Proposicionales

La construcción se basa en dos ideas:

- Se restringe a la búsqueda de un plan de largo **n**
- Se transforma el problema restringido a un problema de satisfacibilidad

El problema de planning como formulas Proposicionales

Cada predicado con K argumentos es transformado en un predicado con K+1 argumentos, donde el ultimo argumento es el paso

at(r1, l1) --------- at(r1,l1, i) donde $0 \leq i \leq \mathbf{n}$

A estos predicados se los denomina f_i

Acciones

move(r1, l1, l2) ------- move(r1,l1,l2, i) donde $0 \leq i \leq \mathbf{n}$-1

A esta forma de ver a las acciones se la denomina a_i

Transformación del problema: Paso 1

- Estado Inicial

f_0 $\quad$ $\sim f_0$

$f \in S_0$ $\quad$ $f \quad S_0$

Transformación del problema: Paso 2

- Estado Final

f_n $\quad$ $\sim f_n$

$f \in g^+$ $\quad$ $f \quad g^-$

Transformación del problema: Paso 3

- Acciones

las precondiciones deben ser verdaderas en el paso i y los efectos mantenerse en el paso i+1
Para cada acción y por cada 0≤ i ≤ **n**

$$a_i \rightarrow \left(\underset{p \in pre(a)}{p_i} \quad \underset{e \in eff(a)}{e_{i+1}} \right)$$

Transformación del problema: Paso 4

- Axiomas

Necesidad de modelar que si algo cambia de un estado al otro sí o sí es por causa de una acción.

$$\sim f_i \quad f_{i+1} \rightarrow \left(\underset{a \in A \mid fi \in eff^{+}(a)}{a_i} \right)$$

$$f_i \quad \sim f_{i+1} \rightarrow \left(\underset{a \in A \mid fi \in eff^{-}(a)}{a_i} \right)$$

$p \in pre(a)$

Transformación del problema: Paso 5

- Sólo una acción ocurre en cada paso

por cada $0 \leq i \leq \mathbf{n}\text{-}1$ y por cada $a_i, b_i \in A$ (distintos ai, bi)

$$\sim a_i \quad \sim b_i$$

Ejemplo

- Ejemplo del robot
  - Estado inicial: El robot r1 esta en l1
  - Objetivo: El robot r2 este en l2
  - Acción disponible:

mover(R, L, L')

pre: at (R, L)

eff: at(R, L'), ~at(R,L)

Transformación

n = 1

(estado inicial) at(r1,l1,0) ~at(r1,l2,0)

(estado final) at(r1,l2,1) ~at(r1,l1,1)

Transformación

- Acción

(mover1) mover(r1,l1,l2,0) → at(r1,l1,0) at(r1,l2,1) ~at(r1,l1,1)

(mover2) mover(r1,l2,l1,0) → at(r1,l2,0) at(r1,l1,1) ~at(r1,l2,1)

Transformación

- Axiomas

(ax1) ~at(r1,l1,0) at(r1,l1,1) → mover(r1,l2,l1,0)
(ax2) ~at(r1,l2,0) at(r1,l2,1) → mover(r1,l1,l2,0)

(ax3) at(r1,l1,0) ~at(r1,l1,1) → mover(r1,l1,l2,0)
(ax4) at(r1,l2,0) ~at(r1,l2,1) → mover(r1,l2,l1,0)

- Exclusión de acciones

~ mover(r1,l1,l2,0) ~ mover(r1,l2,l1,0)

Extracción de la solución

- Dada la secuencia de largo **n** se toma como secuencia de variables proposicionales a
  - **{ $a_i(0)$,…….$a_n(n-1)$}**
  - Donde **$a_i(j)$** es la variable que representa a la acción **a_i** en el paso **j**

Solución en el ejemplo

- $\mathbf{a_1}$ = mover(r1,l1,l2)
- Si se agregan acciones se agregan variables nuevas.

Preguntas