Complejidad

Paula Severi

University of Leicester

Facultad de Ingeniería. Universidad de la República, Montevideo, Uruguay. Noviembre-Diciembre 2018.

Resumen de esta clase

1. Clases de complejidad. Dificultad de un problema de decisión.

2. Satisfactibilidad de conceptos en $\mathcal{ALC}$ es *PSPACE*-difícil.

Complejidad

¿ Cuáles son los recursos computacionales requeridos para resolver un problema (una tarea)?

Problemas de decisión

Es un problema donde las respuestas posibles son SI/NO.

Ejemplos de problema de decisión:

1. Sea n un natural. ¿ n es primo ?
2. Sea C en $\mathcal{ALC}$. ¿ C es satisfactible ?
3. Sea $\mathcal{K}$ una base de conocimientos en $\mathcal{ALC}$. ¿ $\mathcal{K}$ es consistente?

Recursos

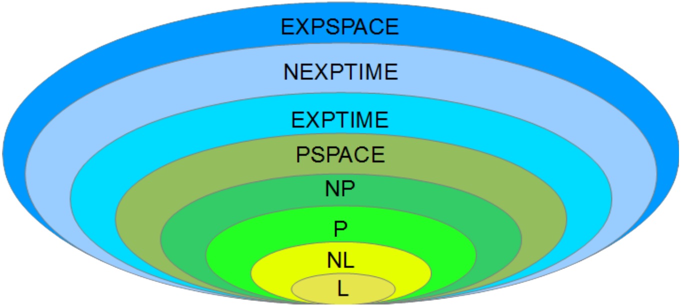
- tiempo: número de pasos de ejecución que un algoritmo emplea para resolver un problema.
- espacio: cantidad de memoria utilizada para resolver el problema.

Clases de Complejidad

- La clase P es el conjunto de problemas de decisión que pueden ser resueltos en tiempo polinómico por una máquina de Turing determinista.

- La clase PSPACE es el conjunto de problemas de decisión que pueden ser resueltos por una máquina de Turing determinista en espacio polinómico.

Clases de Complejidad



Inclusiones estrictas conocidas:

$P \subsetneq EXPTIME$

$NP \subsetneq NEXPTIME$

Dificultad de un problema

PSPACE-difícil

un problema D de decisión es PSPACE-difícil si todo problema en PSPACE puede ser transformado a D en tiempo polinomial.

PSPACE-completo

un problema de decisión D es PSPACE-completo si

1. D está en la clase PSPACE
2. D es PSPACE-difícil

Satisfactibilidad de conceptos en $\mathcal{ALC}$

Problema de decisión

Sea C en $\mathcal{ALC}$. ¿ C es satisfactible ?

Satisfactibilidad de conceptos en $\mathcal{ALC}$ está en *PSPACE*

Ya vimos un algoritmo PSPACE que resuelve este problema.

Satisfactibilidad de conceptos en $\mathcal{ALC}$ es *PSPACE*-difícil

Reducimos un problema sobre juegos que se sabe que es PSPACE a nuestro problema.

Juego booleano finito

$G = (\varphi, \Gamma_1, \Gamma_2)$

1. ϕ fórmula de logica proposicional con variables $\{p_1, \ldots, p_n\}$
2. $\Gamma_1 = \{p_1, p_3, \ldots, p_{n-1}\}$ variables del jugador 1
3. $\Gamma_2 = \{p_2, p_4, \ldots, p_n\}$ variables del jugador 2
4. Dos jugadores se turnan para elegir valores de verdad
   Jugador 1 elige un valor de verdad para p_1
   Jugador 2 elige un valor de verdad para p_2
   Jugador 3 elige un valor de verdad para p_3

Si la asignación de valores de verdad satisface φ entonces el jugador 1 gana. Sino, el jugador 2 gana.

Ejemplo de juego booleano finito

$G = (\varphi, \Gamma_1, \Gamma_2)$

1. $\varphi = (\neg p_1 \rightarrow p_2) \wedge (p_1 \wedge p_2 \rightarrow p_3 \vee p_4) \wedge (\neg p_2 \rightarrow p_4 \rightarrow \neg p_3)$
2. $\Gamma_1 = \{p_1, p_3\}$ variables del jugador 1
3. $\Gamma_2 = \{p_2, p_4\}$ variables del jugador 2

Formen parejas y a jugar!

Estrategia ganadora

Problema de decisión

Dado un juego G booleano finito, ¿ jugador 1 tiene una estrategia ganadora ?

Turnos de jugadores $i = 1, \ldots, n$
Para el jugador 1, i es impar   para el jugador 2, i es par

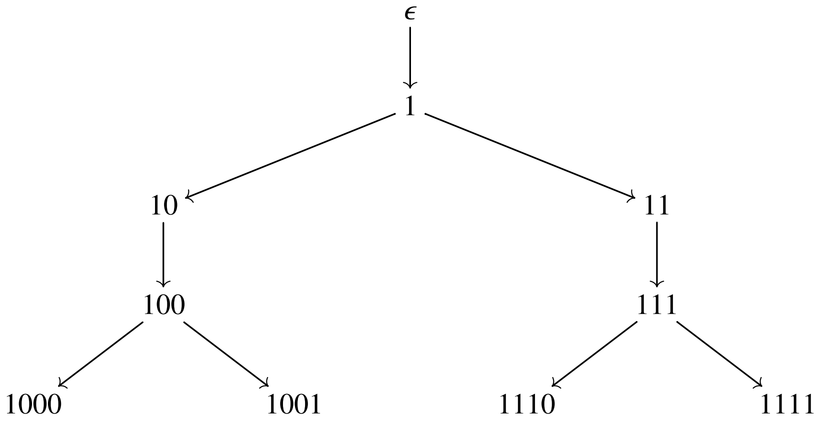
Estrategia ganadora

Es un árbol con nodos etiquetados con palabras en $\{0, 1\}^*$

1. nodos a nivel i tienen etiquetas en $\{0, 1\}^i$
2. nodos a nivel n tienen etiquetas que satisfacen φ
3. nodos a nivel par con etiqueta t tienen un sucesor etiquetado con $t0$ o $t1$
4. nodos a nivel impar con etiqueta t tienen un sucesor etiquetado con $t0$ y otro con $t1$

Ejemplo de estrategia ganadora

$$\varphi = (\neg p_1 \to p_2) \wedge (p_1 \wedge p_2 \to p_3 \vee p_4) \wedge (\neg p_2 \to p_4 \to \neg p_3)$$



Reducción del problema de estrategias al de satisfactibilidad

De $G = (\varphi, \Gamma_1, \Gamma_2)$ construimos C_G en $\mathcal{ALC}$ tal que:

Propiedad

Jugador 1 tiene una estrategia ganadora en G si y solo si C_G es satisfactible.

$C_G = C_1 \sqcap C_2 \sqcap C_3 \sqcap C_4$ con nombres de conceptos $P_1, P_2, \ldots, P_n$

$\hat{\varphi}$ transformación de φ a un concepto en $\mathcal{ALC}$

$\forall R^i.C$ denota $\underbrace{\forall R.\ldots\forall R}_{i \text{ veces}}.C$

Reducción del problema de estrategias al de satisfactibilidad

1. nodos a nivel impar tienen dos sucesores

$$C_1 = \bigsqcap_{i \in \{1,3,\dots,n-1\}} \forall R^i.(\exists R.\neg P_{i+1} \sqcap P_{i+1})$$

2. nodos a nivel par tienen un sucesor

$$C_2 = \bigsqcap_{i \in \{0,2,\dots,n\}} \forall R^i.\exists R.\top$$

3. una vez elegido el valor de verdad, este queda fijo

$$C_3 = \bigsqcap_{1 \le i \le j \le n} \forall R^j.(P_i \to \forall R.P_i) \sqcap (\neg P_i \to \forall R.\neg P_i)$$

4. En las hojas, la fórmula φ es verdadera

$$C_4 = \forall R^n.\hat{\varphi}$$

Cliquear acá para ver complejidades

no está completamente actualizado....