Lógica computacional

Tema: Lógica Clausular Proposicional

Facultad de Ciencias
Universidad Nacional Autónoma de México

Material desarrollado bajo el proyecto UNAM-PAPIME PE102117.

Fnn

Una fórmula φ está en **forma normal negativa** si y sólo si cumple las siguientes condiciones:

1. φ no contiene equivalencias ni implicaciones
2. Las negaciones que figuran en φ afectan sólo a fórmulas atómicas.

La transformación a forma normal negativa se apoya en las siguientes equivalencias:

- Doble Negación: $\neg\neg\varphi \equiv \varphi$.
- De Morgan: $\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$.
- De Morgan: $\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi$.
- $\neg(\varphi \to \psi) \equiv \varphi \wedge \neg\psi$
- $\neg(\varphi \to \psi) \equiv \neg\varphi \to \psi \equiv \varphi \to \neg\psi$.

Literales.

Una literal ℓ es una fórmula atómica (variable proposicional p, $\bot$ o $\top$) o la negación de una fórmula atómica.

Una literal es **negativa** si es una negación, en otro caso decimos que es **positiva**.

Dada una literal ℓ definimos su **literal contraria**, denotada ℓ^c, como sigue:

$$\ell^c = \begin{cases} a & \text{si} \quad \ell = \neg a \\ \neg a & \text{si} \quad \ell = a \end{cases}$$

Donde a es una fórmula atómica, es decir, *varp*, $\top$ o $\bot$.

El par $\{\ell, \ell^c\}$ se llama un par de **literales complementarias.**

FNC.

Una cláusula $\mathcal{C}$ es una literal o una disyunción de literales.

Una fórmula φ está en **forma normal conjuntiva** ($\texttt{fnc}$) si y sólo si es de la forma $\mathcal{C}_1 \wedge \mathcal{C}_2 \wedge \ldots \wedge \mathcal{C}_n$, donde cada $\mathcal{C}_i$ es una cláusula.

Formas Normales

En particular, se sigue que cualquier literal y cualquier cláusula están en forma normal conjuntiva.
Los conceptos anteriores de literal, cláusula y forma normal conjuntiva se definen de manera breve mediante la siguiente gramática:

$$
\begin{array}{rcll}
S & ::= & F & \\
F & ::= & \mathcal{C} \mid \mathcal{C} \wedge F & \text{- - forma normal conjuntiva} \\
\mathcal{C} & ::= & \ell \mid \ell \vee \mathcal{C} & \text{- - cláusulas} \\
\ell & ::= & a \mid \neg\, a & \text{- - literales} \\
a & ::= & \bot \mid \top \mid p & \text{- - atómicas}
\end{array}
$$

Formas Normales

El empleo de las formas normales conjuntivas simplifica el procedimiento para decidir si una fórmula dada es válida, es decir, es tautología.

Una cláusula $\mathcal{C} = \ell_1 \vee \ell_2 \vee \ldots \vee \ell_n$ es tautología ($\models \varphi$) si y sólo si existen $1 \leqslant i, j \leqslant n$ tales que $\ell_i^c = \ell_j$.

Es decir, $\models \mathcal{C}$ si y sólo si $\mathcal{C}$ contiene un par de literales complementarias.

Formas Normales

La proposición anterior permite tener un algoritmo para verificar si $\models \varphi$, cuando φ está en forma normal conjuntiva, digamos $\varphi = \mathcal{C}_1 \wedge \ldots \wedge \mathcal{C}_n$:

1. Para cada $1 \leqslant i \leqslant n$, buscar en $\mathcal{C}_i$ un par de literales complementarias.
2. Si tal par existe para cada cláusula $\mathcal{C}_i$ entonces $\models \varphi$.
3. En otro caso $\not\models \varphi$, es decir φ no es tautología.

Ejercicio 1: Decidir si $p \wedge (p \to r) \to (q \to r)$ es tautología.

Formas Normales

Proposición

- $\models \varphi$ si y sólo si $\neg\varphi$ es no satisfacible.
- φ es satisfacible si y sólo si $\not\models \neg\varphi$

El Problema SAT

La forma más común de enunciar el problema de satisfacibilidad para la lógica proposicional (usualmented enotado como SAT) es el siguiente:

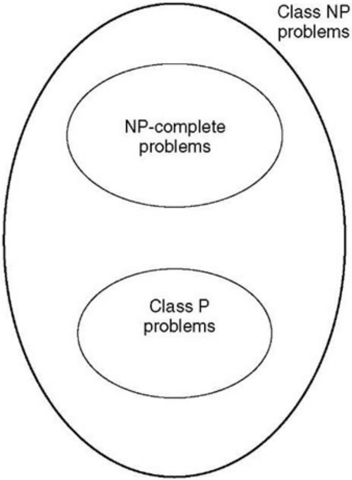
Dado un conjunto $P = \{p_1, \dots, p_n\}$ de variables proposicionales y un conjunto C de cláusulas con variables en P ¿Existe una interpretación $\mathcal{I}$ que satisfaga a C?

El Problema SAT

- SAT fue el primer problema NP-completo conocido (Cook 1971).
- A partir de los años 90, se han desarrollado diversos algoritmos para resolver el problema.
- Conjetura: *Cualquier algoritmo que resuelve SAT es exponencial en el número de variables, en el peor de los casos.*
- El razonamiento automatizado se encarga, entre otras cosas, del desarrollo de algoritmos para resolver el problema SAT.

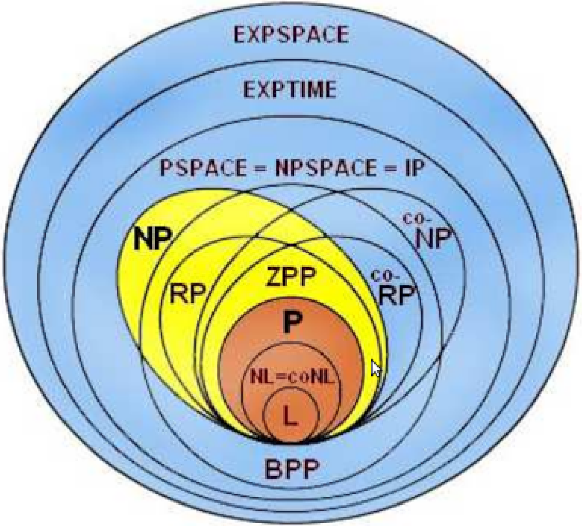
El Problema SAT

Nota sobre complejidad



El Problema SAT

Nota sobre complejidad



The international SAT Competitions web page

Current competition

	SAT 2016 competition					
Organizers	Marijn Heule, Matti Järvisalo Tomáš Balyo					
Proceedings	Descriptions of the solvers and benchmarks					
Benchmarks	Available here					
Solvers	Available here					
	Gold	**Silver**	**Bronze**	**Gold**	**Silver**	
	Agile Track			**Main Track**		
SAT+UNSAT	Riss	TB_Glucose	CHBR_Glucose	MapleCOMSPS	Riss	
	Parallel Track			**No-Limit Track**		
SAT+UNSAT	Treengeling	Plingeling	CryptoMiniSat	BreakIDCOMiniSatPS	Lingeling	
	Best Application Benchmark Solver in the Main Track			**Best Crafted Benchmark Solver in the Main Track**		
SAT+UNSAT	MapleCOMSPS			TC Glucose		

Past competitions

In 2015, we had a SAT-Race 2015!

	SAT 2014 competition					
Organizing committee	Anton Belov, Daniel Diepold, Marijn Heule, Matti Järvisalo					
Judges	Pete Manolios, Lakhdar Sais and Peter Stuckey					
Proceedings	Descriptions of the solvers and benchmarks					
Benchmarks	Application, Hard combinatorial, Random					
Solvers	Source code available in EDACC					
	Application			**Hard combinatorial**		
	Gold	**Silver**	**Bronze**	**Gold**	**Silver**	
				Core solvers		
SAT+UNSAT	Lingeling	SWDiA5BY	Riss BlackBox	glueSplit_clasp	Lingeling	Spa
SAT	minisat_blbd	Riss BlackBox	SWDiA5BY	SparrowToRiss	CCAnr+glucose	
Certified UNSAT	Lingeling (druplig)	glucose	SWDiA5BY	Riss BlackBox	Lingeling (druplig)	
				Core solvers, Parallel		
SAT+UNSAT	Plingeling	PeneLoPe	Treengeling	Treengeling	Plingeling	pr
SAT						
				Minisat hack		
SAT+UNSAT	MiniSat_HACK_999ED	minisat_blbd	ROKKminisat			

Resolución Binaria

Resolución binaria

Sean $\mathcal{C}_1$, $\mathcal{C}_2$ cláusulas y ℓ una literal. La regla de inferencia conocida como **resolución binaria proposicional** se define como sigue:

$$\frac{\mathcal{C}_1 \vee \ell \qquad \ell^c \vee \mathcal{C}_2}{\mathcal{C}_1 \vee \mathcal{C}_2}\ (\mathit{Res})$$

donde ℓ^c es la literal contraria de ℓ. En tal situación decimos que se **resuelven** las dos premisas con respecto a la literal ℓ y a la cláusula resultante $\mathcal{C}_1 \vee \mathcal{C}_2$ se le llama **resolvente o resolvente binario**.

Resolución Binaria

Si bien en la definición de la regla las literales ℓ y ℓ^c aparecen al final y principio de las cláusulas respectivamente, el orden no importa dado que la disyunción es conmutativa.

Por ejemplo:

$$\frac{\neg p \vee q \vee \mathbf{r} \qquad s \vee \neg \mathbf{r} \vee \neg t}{\neg p \vee q \vee s \vee \neg t} \qquad \frac{t \vee \neg \mathbf{s} \vee q \qquad \neg q \vee w \vee \mathbf{s} \vee u}{t \vee q \vee \neg q \vee w \vee u}$$

Resolución Binaria

Dado que las literales también son cláusulas la aplicación de resolución a p y $\neg p$ devuelve como resultado la llamada **clásula vacía**, denotada $\square$, es decir, la siguiente es una instancia válida de resolución:

$$\frac{p \quad \neg p}{\square}$$

Resolución Binaria

La resolución binaria proporciona un método de decisión para la lógica que utiliza el principio de refutación para decidir la consecuencia lógica:

para decidir si $\Gamma \models \varphi$ basta demostrar que $\Gamma \cup \{\neg\varphi\}$ es insatisfacible, para lo cual basta con obtener la cláusula vacía usando la regla de resolución binaria, a partir del conjunto de cláusulas de la formas normales conjuntivas del conjunto $\Gamma \cup \{\neg\varphi\}$.

Este proceso se conoce como una **refutación** del conjunto de cláusulas.

Algoritmo de Saturación

n-ésima resolución

Si $\mathbb{S}$ es cualquier conjunto de cláusulas, entonces la resolución de $\mathbb{S}$, denotada $\mathcal{R}(\mathbb{S})$, es el conjunto que consiste de $\mathbb{S}$ junto con todos los resolventes de cláusulas de $\mathbb{S}$, es decir:

$\mathcal{R}(\mathbb{S}) = \mathbb{S} \cup \{E \mid \text{existen } C, D \in \mathbb{S} \text{ tales que } E \text{ es un resolvente de } C \text{ y } D\}$.

La n-ésima resolución de $\mathbb{S}$ se define recursivamente como sigue:

$$
\begin{aligned}
Res_0(\mathbb{S}) &= \mathbb{S} \\
Res_{n+1}(\mathbb{S}) &= \mathcal{R}(Res_n(\mathbb{S}))
\end{aligned}
$$

Algoritmo de Saturación

n-ésima resolución

Sea $\mathbb{S}$ es un conjunto finito de cláusulas:

$\mathbb{S}$ es no satisfacible si y sólo si $\square \in Res_n(\mathbb{S})$ para alguna $n \in \mathbb{N}$.

Un **algoritmo de saturación** se encargan de generar todos los posibles resolventes a partir de un conjunto dado $\mathbb{S}$.

Para verificar si un conjunto de cláusulas $\mathbb{S}$ es **insatisfacible**, basta construir con un algoritmo de saturación los conjuntos $Res_n(\mathbb{S})$ hasta hallar $\square$.

Algoritmo de Saturación

Analicemos los escenarios posibles durante la ejecución de un algoritmo de saturación :

1. En algún momento $\square$ es generada, es decir $\square \in Res_n(\mathbb{S})$ para algún $n \in \mathbb{N}$.
   En este caso el conjunto Γ de entrada es insatisfacible.

2. El algoritmo termina sin generar $\square$ jamás, es decir, en algún momento se tiene $Res_n(\mathbb{S}) = Res_{n+1}(\mathbb{S})$ por lo que no hay más resolventes posibles, pero $\square \notin Res_n(\mathbb{S})$.
   En este caso Γ es satisfacible.