IIC2213 — Lógica para Ciencia de la Computación

# Apuntes Semana 13

## 1. VAL y SAT

Una $L$-fórmula $\varphi$ se dice *satisfacible* si existe una estructura $\mathcal{A}$ y una asignación $\tau$ tal que $(\mathcal{A}, \tau) \models \varphi$.

Una $L$-fórmula $\varphi$ se dice *válida* si para toda estructura $\mathcal{A}$ y toda asignación $\tau$ se tiene que $(\mathcal{A}, \tau) \models \varphi$ (las fórmulas válidas son el equivalente a las tautologías en lógica proposicional).

Notar que si $\varphi$ es una $L$-oración (sin variables libres) entonces $\varphi$ es satisfacible si existe $L$-estructura $\mathcal{A}$ tal que $\mathcal{A} \models \varphi$ y $\varphi$ es válida si para toda $L$-estructura $\mathcal{A}$ se tiene que $\mathcal{A} \models \varphi$.

**Teorema (Church)**. El siguiente lenguaje es indecidible:

$$\text{VAL} = \{\varphi \mid \varphi \text{ es una oración válida }\}.$$

La demostración no es parte de los contenidos base de este curso, aunque como referencia la incluimos al final de las notas. Lo que si puedes hacer es usar el teorema de Church para demostrar el siguiente corolario:

**Ejercicio**. Muestra que el siguiente lenguaje es indecidible:

$$\text{SAT} = \{\varphi \mid \varphi \text{ es una oración satisfacible }\}.$$

**Ejercicio**. Muestra que VAL es recursivamente enumerable y que SAT es coRE.

## 2. Aplicación: teoría de grafos

Sea $L = \{E(\cdot, \cdot)\}$ el vocabulario sobre grafos. Usamos STRUCT$[L]$ para denotar a todas las estructuras definibles con un vocabulario: en este caso, STRUCT$[L]$ son todos los grafos.

Una propiedad sobre grafos es un subconjunto $P \subseteq \text{STRUCT}[L]$ de grafos. Decimos que la propiedad es *definible* en lógica de primer orden si existe una $L$-oración $\varphi$ tal que $\mathcal{A} \models \varphi$ si y solo si $A \in P$, es decir, un grafo $\mathcal{A}$ satisface a $\varphi$ si y solo si $\mathcal{A}$ pertenece a la propiedad.

Muestra que las siguientes propiedades sobre grafos son definibles en lógica de primer orden:

- El grafo es un clique
- El grafo tiene al menos 5 nodos

- El grafo tiene al menos 3 aristas
- El grafo no tiene triángulos

Considera ahora el vocabulario $L' = \{E(\cdot,\cdot), a, b\}$ que ademas incorpora dos constantes. En este caso STRUCT[$L'$] son todos los grafos con dos de sus nodos distinguidos: un nodo es la interpretación de la constante $a$ y el otro es la interpretación de la constante $b$.

Muestra que las siguientes propiedades son definibles en lógica de primer orden usando $L'$:

- Hay una arista entre la interpretación de $a$ y la interpretación de $b$.
- El único nodo con más de 2 vecinos es el nodo correspondiente a la interpretación de $a$.

# 3. Consecuencia lógica y sistemas deductivos

Al igual que con lógica proposicional, definimos el operador de consecuencia lógica: Para dos oraciones $\varphi$ y $\psi$ sobre un vocabulario $L$, $\varphi \models \psi$ si y solo si toda $L$-estructura $\mathcal{A}$ que satisface a $\varphi$ también satisface a $\psi$.

De la misma forma, si $\varphi$ y $\psi$ tienen las mismas variables libres, $\varphi \models \psi$ si y solo si toda $L$-estructura $\mathcal{A}$ y asignación $\tau$ que satisface a $\varphi$ también satisface a $\psi$.

## 3.1. El sistema deductivo de Hilbert

El sistema deductivo de Hilbert es un sistema para verificar cuando $\varphi \models \psi$. Consta de dos reglas, y un conjunto de esquemas para generar fórmulas válidas. Las reglas son las siguientes:

**Reglas.**

- Modus Ponens:

$$\frac{\varphi \to \psi \quad \varphi}{\psi}$$

- Generalización: Si $y$ no aparece libre en $\varphi$, entonces

$$\frac{\varphi \to \psi(y)}{\varphi \to \forall x \psi(x)}$$

**Axiomas válidos.** Con respecto a las fórmulas válidas, Para fórmulas arbitrarias $\varphi$, $\psi$, $\theta$, nota que todas estas fórmulas son válidas:

1. $\varphi \to (\psi \to \varphi)$
2. $(\varphi \to (\psi \to \theta)) \to ((\varphi \to \psi) \to (\varphi \to \theta))$
3. $(\neg\varphi \to \neg\psi) \to ((\neg\varphi \to \psi) \to \varphi)$
4. $(\forall x\, \varphi(x)) \to \varphi(t)$, donde $t$ es un término cualquiera
5. $\varphi(t) \to (\exists x\, \varphi(x))$, donde $t$ es un término cualquiera
6. $(\exists x\, \varphi) \leftrightarrow (\neg\forall x\, \neg\varphi)$
7. $\forall x\, (x = x)$
8. $\forall x \forall y\, (x = y \to y = x)$
9. $\forall x \forall y \forall z\, ((x = y \wedge y = z) \to x = z)$
10. Para todo predicado $m$-ario $P$:

$$\forall x_1 \cdots \forall x_m \forall y_1 \cdots \forall y_m\, ((P(x_1, \ldots, x_m) \wedge x_1 = y_1 \wedge \cdots \wedge x_m = y_m) \to P(y_1, \ldots, y_m))$$

**Funcionamiento.**

Dado un conjunto de fórmulas $\Sigma \cup \{\varphi\}$, una deducción formal de $\varphi$ desde $\Sigma$ mediante el Sistema de Hilbert es una secuencia de fórmulas $\varphi_1$, $\varphi_2$, …, $\varphi_n$ tal que:

- Para cada $i \leq n$:
  - $\varphi_i \in \Sigma$ o
  - $\varphi_i$ es un axioma válido, construido según el esquema de arriba, o
  - existen $j, k < i$ tales que $\varphi_i$ es obtenido desde $\varphi_j$ y $\varphi_k$ usando modus ponens o
  - existe $j < i$ tal que $\varphi_i$ es obtenido desde $\varphi_j$ usando la regla de generalización.
- $\varphi_n = \varphi$

Usamos $\Sigma \vdash \varphi$ para denotar que puede deducirse $\varphi$ desde $\Sigma$ usando el sistema deductivo de Hilbert.

**Propiedades.** Tal como en el caso de resolución para la lógica proposicional, podemos mostrar que el sistema deductivo de Hilbert es correcto y completo:

**Teorema (Correctitud).** Si $\Sigma \vdash \varphi$, entonces $\Sigma \models \varphi$.

**Teorema (Compleitud de Gödel).** Si $\Sigma \models \varphi$, entonces $\Sigma \vdash \varphi$.

## 3.2. El teorema de Compacidad

Un corolario de tener un sistema deductivo correcto y completo como el que mostramos arriba, es que la lógica de primer orden satisface la propiedad de *Compacidad*, en el siguiente sentido.

Decimos que un conjunto $\Sigma$ de fórmulas es finitamente satisfacible si cada subconjunto finito de $\Sigma$ es satisfacible.

**Teorema (compacidad)**. Un conjunto de fórmulas $\Sigma$ es satisfacible si y sólo si $\Sigma$ es finitamente satisfacible.

**Ejercicio**. Demostrar el teorema (podemos hacerlo usando la correctitud y completitud del sistema de Hilbert)

Este teorema solo hace sentido cuando el conjunto $\Sigma$ de oraciones es *infinito*. ¿Podemos hacer eso? ¡Claro! Y podemos tener también estructuras infinitas. Consider por ejemplo, sobre un vocabulario $\{E\}$, para cada $n > 0$, la fórmula $\psi_n = \exists x_1, \ldots, x_n \bigwedge_{i \neq j} \neg(x_i = x_j)$, que dice que existen $n$ elementos distintos en el dominio, y sea $\Sigma = \{\psi_n \mid n > 0\}$. Este es un conjunto infinito de fórmulas.

¿Es $\Sigma = \{\psi_n \mid n > 0\}$ satisfacible? Podemos usar el teorema de compacidad, y analizar que pasa con cada subconjunto finito de $\Sigma$. Sea, en efecto, $S \subsetneq \Sigma$ un subconjunto finito arbitrario. Como $S$ es finito, $S$ contiene a una cierta cantidad de fórmulas $\psi_i$, y en efecto hay un entero $k$ tal que $\psi_k$ es la fórmula con el índice más grande en $S$. Luego $\psi_k = \exists x_1, \ldots, x_k \bigwedge_{i \neq j} \neg(x_i = x_j)$, y por tanto cualquier estructura $\mathcal{A}$ con $k$ o más elementos satisface a $\varphi_k$, y satisface de hecho a cualquier $\varphi_\ell$ con $\ell \leq k$, por lo que el conjunto $S$ es satisfacible.

Dado que todo subconjunto finito arbitrario de $S$ es satisfacible, por el teorema de compacidad tenemos que $\Sigma = \{\psi_n \mid n > 0\}$ es satisfacible. En efecto, puedes comprobar que cualquier grafo infinito (como estructura sobre el vocabulario $\{E\}$ satisface a $\Sigma$.

## 3.3. Usando compacidad para mostrar que ciertas propiedades no son definibles

Sea $L = \{E\}$ el vocabulario de grafos. Ahora podemos mostrar que la siguiente propiedad $F$ no es definible:

$$F = \{\mathcal{A} \in \text{STRUCT}[L] \mid \mathcal{A} \text{ tiene un dominio finito } \}$$

Asumamos para llegar a una contradicción que esa propiedad es definible por una oración $\varphi$, es decir, tal que una estructura $\mathcal{A}$ pertenece a $F$ si y solo si $\mathcal{A} \models \varphi$, y recordemos el conjunto $\Sigma = \{\psi_n \mid n > 0\}$ definido arriba. Entonces tenemos que $\{\varphi\} \cup \Sigma$ no es satisfacible, dado que las únicas estructuras que satisfacen $\Sigma$ tienen dominio infinito, no hay ninguna estructura que pertenezca a $F$ y satisfaga a $\Sigma$ al mismo tiempo.

Sin embargo, todo subconjunto finito de $\{\varphi\} \cup \Sigma$ es satisfacible. En efecto, ya demostramos que todo subconjunto finito de $\Sigma$ es satisfacible, y en efecto es satisfacible por una estructura

con dominio finito (ver la demostración arriba). Luego, cualquier subconjunto finito de $\Sigma$ también es satisfacible junto a $\varphi$.

De estos dos hechos obtenemos una contradicción por el teorema de compacidad: si todo subconjunto finito de $\{\varphi\} \cup \Gamma$ es satisfacible, entonces $\{\varphi\} \cup \Gamma$ debería ser satisfacible.

Como obtenemos una contradicción, probamos que $F$ no puede ser definible por ningún conjunto finito de oraciones.

## 3.4. Ejercicios propuestos

**Ejercicio**. Usa la técnica de arriba para mostrar que estas propiedades no son definibles en LPO sobre el vocabulario $\mathcal{L} = \{a, b, E(\cdot,\cdot), R(\cdot,\cdot)\}$:

- $P_5 = \{\mathcal{A} \in \text{STRUCT}[L] \mid R^{\mathcal{A}}$ contiene a la clausura transitiva de $E^{\mathcal{A}}\}$
- $P_6 = \{\mathcal{A} \in \text{STRUCT}[L] \mid E^{\mathcal{A}}$ es un grafo con un camino Euleriano $\}$ (Los grafos con caminos eulerianos son aquellos en los que todos sus vértices tienen grado par)
- $P_7 = \{\mathcal{A} \in \text{STRUCT}[L] \mid a^{\mathcal{A}}$ está conectado a $b^{\mathcal{A}}$ mediante $E^{\mathcal{A}}\}$

# 4. Material extra

Este material lo dejamos como referencia, no alcanzamos a verlo como para evaluarlo.

## Demostración del teorema de Church

La demostración es por una reducción desde el siguiente lenguaje:

$$L_\varepsilon = \{w \in \{0,1\}^* \mid \text{ existe máquina de Turing } M \text{ tal que } w = C(M) \text{ y } M \text{ acepta al string vacío } \varepsilon\}$$

**Ejercicio**. Muestra que el lenguaje $L_\varepsilon$ es indecidible.

Continuando, para la reducción mostramos como construir, para cada máquina de turing $M$, una fórmula $\varphi_M$ tal que $\varphi_M$ es válida si y solo si $M$ acepta a $\varepsilon$. Suponemos que $M = (Q, \{0,1\}, q_0, \delta, F)$, donde

- $Q = q_1, \ldots, q_m$.
- $F = \{q_m\}$ (la máquina tiene un solo estado final).
- No existe transición en $\delta$ para $q_m$ (el estado final no tiene transiciones definidas).

¿Puedes explicar por qué estas suposiciones son sin pérdida de generalidad?

Definimos un vocabulario con $m + 6$ relaciones: $\{P, C, U, B, T, L\} \cup \{E_i \mid 1 \leq i \leq m\}$. De esas $P$ y cada $E_i$ son unarias, el resto son binarias. Intuitivamente, la interpretación que daremos a estas relaciones será la siguiente:

- $P(t)$ solo si $t$ es el tiempo de partida de la máquina.
- $C(t, p)$: $M$ tiene un 0 en la posición $p$ en el tiempo $t$.
- $U(t, p)$: $M$ tiene un 1 en la posición $p$ en el tiempo $t$.
- $B(t, p)$: $M$ tiene un $B$ en la posición $p$ en el tiempo $t$.
- $E_i(t)$: El estado de la máquina en el tiempo $t$ es $q_i$ $(1 \leq i \leq m)$.
- $T(t, p)$: La cabeza de $M$ está en la posición $p$ en el tiempo $t$.
- $L(x, y)$: Orden lineal total en el dominio.

La idea es, entonces, construir $\varphi_M$ de forma que las estructuras que satisfagan $\varphi$ representen la ejecución de $M$ con input $\varepsilon$.

Definimos $\varphi_M = (\varphi_P \wedge \varphi_L \wedge \varphi_I \wedge \varphi_C \wedge \varphi_\delta) \rightarrow \varphi_A$, donde:

- $\varphi_P$ define que hay un único tiempo de partida:

$$\varphi_P = \exists x(P(x) \wedge (\forall y P(y) \rightarrow x = y))$$

- $\varphi_L$ se asegura que $L$ se interprete como un orden lineal donde cada elemento tiene un antecesor y un sucesor. Se define como la conjunción de las fórmulas:

- $\forall x \neg L(x, x)$;
- $\forall x \forall y \forall z(L(x, y) \wedge L(y, z) \rightarrow L(x, z))$;
- $\forall x \forall y(x = y \vee L(x, y) \vee L(y, x))$;
- $\forall x \exists y(L(x, y) \wedge \neg \exists z(L(x, z) \wedge L(z, y)))$;
- $\forall x \exists y(L(y, x) \wedge \neg \exists z(L(y, z) \wedge L(z, x)))$;

- $\varphi_I$ se asegura que en el tiempo inicial la cinta de la máquina se ve como la configuración inicial de $M$, y que la máquina está en estado $q_0$:

$$\varphi_I = \forall x(P(x) \rightarrow (\forall y B(x, y) \wedge E_0(x) \wedge T(x, x)))$$

- $\varphi_C$ se asegura que la máquina funciona correctamente. Es la conjunción de las siguientes cuatro fórmulas:

- Una fórmula para definir que cada celda tiene un único símbolo en un tiempo dado:

$$\forall x \forall y \big((B(x, y) \wedge \neg U(x, y) \wedge \neg C(x, y)) \vee \\ (\neg B(x, y) \wedge U(x, y) \wedge \neg C(x, y)) \vee (\neg B(x, y) \wedge \neg U(x, y) \wedge C(x, y))\big)$$

- Una formula que asegura que la máquina tiene un único estado:

$$\forall x\Big(\bigvee_{1 \leq i \leq m} \big(E_i(x) \wedge \bigwedge_{j \neq i} \neg E_j(x)\big)\Big)$$

- Otra que asegura que la cabeza siempre está en una única posición:

$$\forall x \exists y\big(T(x, y) \wedge \forall z(T(x, z) \rightarrow z = y)\big)$$

- Y una última que fuerza a que el único contenido de la cinta que puede cambiar de un instante a otro es el apuntado por la cabeza. Usamos $s(x, y)$ como abreviación para $s(x, y) = L(x, y) \wedge \neg \exists z(L(x, z) \wedge L(z, y))$ (recuerda que $T(t, p)$ indica que la cabeza apunta a la posición $p$ en el tiempo $t$):

$$\forall x \forall y \forall z\big((\neg T(x, y) \wedge s(x, z)) \rightarrow \\ ((C(x, y) \wedge C(z, y)) \vee (U(x, y) \wedge U(z, y)) \vee (B(x, y) \wedge B(z, y)))\big)$$

- $\varphi_\delta$ se asegura que los cambios en la cinta de un instante al siguiente son los indicados por la función de transición. $\varphi_\delta$ se define como la conjunción de una fórmula para cada transición en $\delta$, fórmula que intutivamente gobierna los cambios con respecto a esa transición.

Por ejemplo si $\delta(q_i, 0) \to (q_i, 1, \to)$, la fórmula para esa transición sería

$$\forall x \forall y \forall u \forall v \big( (E_i(x) \wedge T(x,y) \wedge C(x,y) \wedge s(x,u) \wedge s(y,v)) \to (E_j(u) \wedge T(u,v) \wedge U(u,y)) \big)$$

- Finalmente, $\varphi_A$ especifica que la máquina acepta (notar que como $q_m$ no tiene transiciones, la máquina inevitablemente acepta si llega a $q_m$.

$$\varphi_A = \exists x \exists y \big( (P(x) \wedge E_m(x)) \vee (P(x) \wedge L(x,y) \wedge E_m(y)) \big)$$

**Ejercicio**. Muestra que $\varphi_M$ es válida si y solo si $M$ acepta a $\varepsilon$.

## Fragmentos decidibles

Usualmente, cuando nos encontramos con problemas indecidibles nos interesa encontrar fragmentos decidibles, que nos permitan encontrar algoritmos para resolver el problema en algunos casos limitados. De ahí la pregunta natural: existen fragmentos importantes en LPO cuyos problemas de VAL y SAT son decidibles?

Vamos a ver un ejemplo, para el que necesitamos un poco de notación. Dada una expresión regular $\alpha$ sobre el alfabeto $\{\forall, \exists\}$, decimos que una $L$-oración $\varphi$ pertenece a la clase $\alpha$ si (1) $\varphi$ es de la forma $Q_1 x_1 \cdots Q_n x_n \ \psi$, con $Q_i \in \{\forall, \exists\}$ y $\psi$ una fórmula sin cuantificadores, y (2) la palabra $Q_1 \cdots Q_n$ pertenece a $\alpha$.

Por ejemplo,

$$\forall x \exists y \exists z R(x,y,z) \qquad \text{y} \qquad \forall x \forall y \forall z (S(x,y) \wedge \neg T(y,z))$$

pertenecen a la clase $\forall^* \exists^*$, pero

$$\forall x \exists y \forall z R(x,y,z) \qquad \text{y} \qquad (\exists x (R(x)) \wedge (\forall x \exists y (x = y))$$

no pertenecen a la clase $\forall^* \exists^*$.

**Teorema (Bernays-Schöfinkel)**. El lenguaje

$$\text{SAT} = \{\varphi \mid \varphi \text{ es una oración en la clase } \exists^* \forall^* \text{ y } \varphi \text{ es satisfacible}\}$$

es decidible y pertenece a $\text{DTIME}(2^{2^n})$ y a $\text{NTIME}(2^n)$.

**Ejercicio**. ¿Puedes mostrar que SAT es decidible para fórmulas en esa clase? Cómo funcionaría un algoritmo en $\text{NTIME}(2^n)$?