Teoría de Modelos Finitos: Motivación

Poder expresivo de una lógica: Caso finito

Desde ahora en adelante nos vamos a concentrar en las estructuras **finitas.**

- Estructuras que tienen dominio finito.

Cambio en la notación: Dado un vocabulario $\mathcal{L}$, struct$[\mathcal{L}]$ es el conjunto de todas las $\mathcal{L}$-estructuras finitas.

Poder expresivo de una lógica: Caso finito

Una propiedad $\mathcal{P}$ de las $\mathcal{L}$-estructuras finitas es un subconjunto de struct$[\mathcal{L}]$.

- Ejemplo: El conjunto de las $\mathcal{L}$-estructuras finitas con dos elementos en el dominio.

Decimos que una propiedad $\mathcal{P}$ es expresable en lógica de primer orden si existe una $\mathcal{L}$-oración φ tal que para toda $\mathfrak{A} \in$ struct$[\mathcal{L}]$:

$$\mathfrak{A} \in \mathcal{P} \quad \text{si y sólo si} \quad \mathfrak{A} \models \varphi.$$

Teorema de compacidad: Caso finito

Sea $\mathcal{L} = \{E(\cdot,\cdot),\ a,\ b\}$, donde a y b son constantes, y $\mathcal{P}$ el conjunto de todas las $\mathcal{L}$-estructuras finitas que tienen un camino entre a y b.

Usando el teorema de compacidad, demostramos que $\mathcal{P}$ no es expresable en primer orden si consideramos estructuras tanto finitas como infinitas.

- ¿Es la demostración válida para el caso finito?
- ¿Cómo debería enunciarse el teorema de compacidad para que la demostración fuera válida? ¿Es esta versión del teorema cierta?

Teorema de compacidad: Caso finito

Dado: Conjunto Σ de oraciones.

Notación: Decimos que Σ tiene un modelo finito si existe una estructura finita que satisface Σ.

Versión "finita" del teorema de compacidad: Si cada subconjunto finito de Σ tiene un modelo finito, entonces Σ tiene un modelo finito.

- ¿Es esta versión cierta?

Versión "finita" del teorema de compacidad: Primer intento

Vamos a tratar de usar completidad para demostrar compacidad ...

Dado	:	Conjunto de oraciones $\Sigma \cup \{\varphi\}$.
Notación	:	$\Sigma \models_F \varphi$ si cada modelo finito de Σ es también modelo de φ

¿Es el sistema de Hilbert completo para el caso finito?

- Vale decir: ¿Es cierto que $\Sigma \models_F \varphi$ implica que $\Sigma \vdash \varphi$?

Versión "finita" del teorema de completidad

Sea $\mathcal{L} = \{R(\cdot,\cdot)\}$. La siguiente oración muestra que el sistema de Hilbert no es completo para el caso finito:

$$\neg\big(\forall x\exists y(R(x,y)) \wedge \forall x\forall y\forall z\,((R(x,y)\wedge R(x,z)) \rightarrow y = z)\wedge$$
$$\forall x\forall y\forall z\,((R(y,x)\wedge R(z,x)) \rightarrow y = z) \wedge \exists y\forall x(\neg R(x,y))\big).$$

¿Por qué?

No podemos usar el sistema de Hilbert para demostrar compacidad para el caso finito.

- Lo anterior no demuestra que no exista un sistema de deducción completo para el caso finito. Esto es tema para más adelante ...

Versión "finita" del teorema de compacidad: Segundo intento

Vamos a demostrar que la versión finita del teorema de compacidad no es cierta. Sean

$$\begin{aligned} \psi_k &= \exists x_1 \ldots \exists x_k \Big(\bigwedge_{1 \leq i < j \leq k} \neg(x_i = x_j) \Big), \\ \Sigma &= \{\psi_k \mid k \geq 1\}. \end{aligned}$$

Cada subconjunto finito de Σ tiene un modelo finito, pero Σ no tiene un modelo finito.

El caso finito ...

Para estudiar el caso de los modelos finito, necesitamos herramientas nuevas.

Pero antes de desarrollar esta herramientas, hay que responder una pregunta natural: **¿Por qué nos interesa tanto el caso finito?**

- Veamos dos aplicaciones: Bases de datos y verificación formal de software.

Más adelante en el curso vamos a ver otras aplicaciones.

Primera aplicación: Bases de datos

Tabla *Vuelo* almacena información sobre vuelos directos:

Partida	Llegada
Santiago	Toronto
Santiago	Lima
Lima	Los Angeles
Lima	Toronto
Lima	Santiago
Toronto	Los Angeles

Primera aplicación: Bases de datos

¿Qué lenguaje usamos para consultar bases de datos?

- Lenguaje estándar: Algebra relacional.

Algebra relacional tiene operadores: selección (σ), proyección (π), join ($\bowtie$), unión ($\cup$) y diferencia ($-$).

¿Por qué se utiliza el álgebra relacional como lenguaje de consulta?

Primera aplicación: Bases de datos

¿Cómo podemos estudiar el poder expresivo del álgebra relacional?

- ¿Por qué necesitamos estudiar esto?

Un resultado fundamental (Codd): Algebra relacional = lógica de primer orden.

Primera aplicación: Bases de datos

Veamos como escribir algunas consultas en lógica de primer orden:

1. ¿Hay un vuelo directo entre Santiago y Lima?

$$\mathit{Vuelo}(\text{Santiago}, \text{Lima}).$$

2. ¿Cuál es la lista de ciudades a las que se puede llegar sin escala desde Santiago?

$$\mathit{Vuelo}(\text{Santiago}, x).$$

3. ¿Hay un vuelo con escala entre Santiago y Los Angeles?

$$\exists x \, (\, \mathit{Vuelo}(\text{Santiago}, x) \wedge \mathit{Vuelo}(x, \text{Los Angeles})).$$

Primera aplicación: Bases de datos

4. ¿Cuál es la lista de ciudades a las que se puede llegar con dos escalas desde Santiago?

$$\exists y \exists z \, (\mathit{Vuelo}(\text{Santiago}, y) \wedge \mathit{Vuelo}(y, z) \wedge \mathit{Vuelo}(z, x)).$$

5. **¿Cuál es la lista de ciudades a las que se puede llegar desde Santiago?**

   Necesitamos recursión.

6. **¿Hay más vuelos desde Santiago que desde Lima?**

   Necesitamos la habilidad de contar.

Bases de datos: Problemas a resolver

¿Podemos demostrar que la lógica de primer orden no tiene recursión?

¿Podemos demostrar que la lógica de primer orden no puede contar?

¿Cómo podemos agregar estas habilidades a la lógica de primer orden?

¿Son estas dos habilidades independientes?

- ¿Si tenemos una lógica con recursión podemos contar?
- ¿Si tenemos una lógica con la habilidad de contar entonces tenemos recursión?

Queremos desarrollar herramientas que nos permitan solucionar estos problemas ...

Segunda aplicación: Verificación formal de software

Ejemplo: Queremos verificar si funciona correctamente el software diseñado para manejar un semáforo.

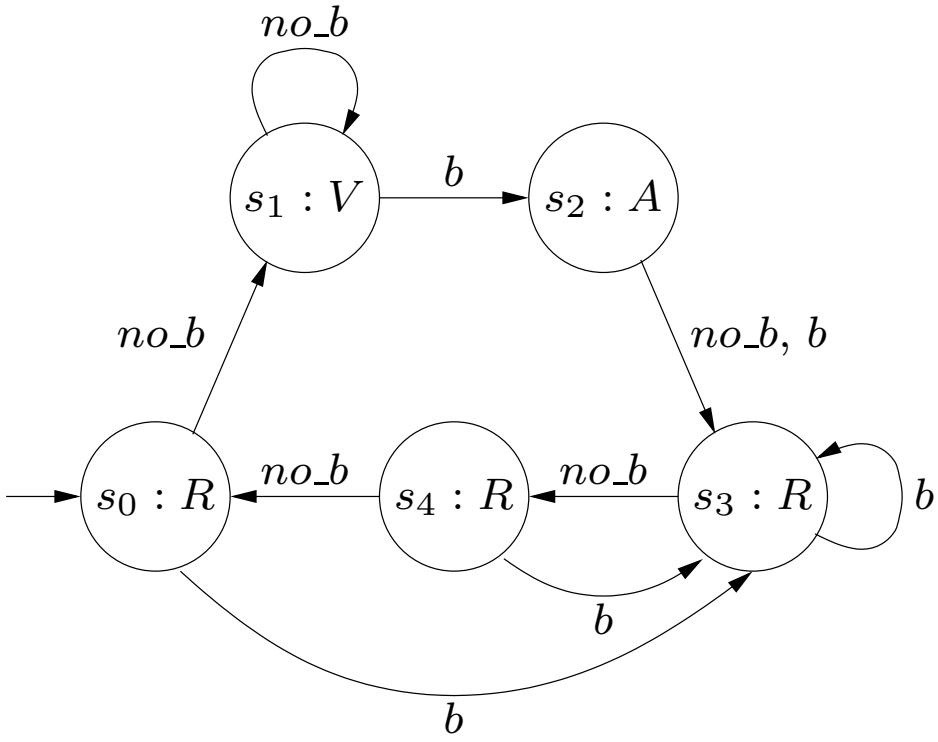
- El semáforo tiene tres estados: R (rojo), A (amarillo) y V (verde), que indican cuál es el color que ven los automovilistas.

- El semáforo además tiene un botón que permite a los peatones ponerlo en rojo.

Para analizar un programa, generalmente se utiliza alguna herramienta que permite *abstraer* el funcionamiento del código.

Veamos que pasa en el caso del semáforo ...

Para el caso del semáforo, se obtiene lo siguiente:



Representamos el comportamiento del programa como un autómata $\mathcal{A}$ con etiquetas en los nodos.

Verificación formal de software

¿Cómo podemos verificar si un programa funciona bien?

El significado de *funcionar bien* es dado por el usuario.

- El usuario debe indicar cuáles son las propiedades que el programa debería satisfacer.
- La satisfacción de estas propiedades debería ser verificada de manera automática.

Necesitamos un lenguaje para poder expresar propiedades de programas (o de sus versiones abstractas).

Verificación formal de software: Lógica temporal

Para expresar propiedades de programas se utiliza lógica temporal.

- Vamos a ver un ejemplo: Linear Temporal Logic (LTL).

φ es una fórmula en LTL sobre un alfabeto Σ si:

- $\varphi = a$ y $a \in \Sigma$.
- $\varphi = (\neg\psi)$ y ψ es una fórmula en LTL.
- $\varphi = (\psi \star \theta)$, $\star \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$ y ψ, θ son fórmulas en LTL.
- $\varphi = (\mathbf{X}\,\psi)$ y ψ es una fórmula en LTL.
- $\varphi = (\psi\,\mathbf{U}\,\theta)$ y ψ, θ son fórmulas en LTL.

LTL: Semántica

X y **U** son conectivos temporales:

- $\mathbf{X}\,\varphi$: En el siguiente estado se cumple φ.
- $\varphi\,\mathbf{U}\,\psi$: φ se cumple hasta que se cumple ψ.

Para definir la semántica de LTL tenemos que hablar de caminos.

Una secuencia infinita $\pi = q_1 q_2 q_3 \cdots$ es un camino en un autómata si para cada i: q_i es un estado y existe un arco de q_i a q_{i+1}.

LTL: Caminos

En el autómata del semáforo son caminos:

$$s_0 s_1 \cdots s_1 \cdots$$

$$s_1 \cdots s_1 \cdots$$

$$s_0 s_1 s_2 s_3 s_4 s_0 s_1 s_2 s_3 s_4 \cdots$$

¿Qué ejecuciones representan estos caminos?

En el autómata del semáforo no es un camino $s_0 s_0 s_1 \cdots s_1 \cdots$.

LTL: Semántica

Una fórmula LTL se evalúa en una posición de un camino.

Dado: LTL fórmula φ y un camino π sacado desde un autómata.

Decimos que φ es satisfecha en la posición i de π, denotado como $\pi, i \models \varphi$, si:

- $\varphi = a$ y la etiqueta de s_i en el autómata es a.
- $\varphi = (\neg\psi)$ y no es verdad que $\pi, i \models \psi$.
- $\varphi = (\psi \wedge \theta)$, $\pi, i \models \psi$ y $\pi, i \models \theta$.
- $\varphi = (\mathbf{X}\,\psi)$ y $\pi, i+1 \models \psi$.
- $\varphi = (\psi\,\mathbf{U}\,\theta)$ y existe $k \geq i$ tal que $\pi, k \models \theta$ y para todo $i \leq j < k$ se tiene que $\pi, j \models \psi$.

LTL: Ejemplos

Sea $\pi = s_0 s_1 \cdots s_1 \cdots s_1$ y considere el autómata del semáforo.

1. ¿Cuáles de las siguientes afirmaciones son ciertas?

$$\begin{aligned} \pi, 1 &\models V \\ \pi, 1 &\models R \\ \pi, 1 &\models \mathbf{X}\, V \\ \pi, 2 &\models R \\ \pi, 2 &\models \mathbf{X}\, \mathbf{X}\, \neg V \end{aligned}$$

2. Sea $\top$ una tautología cualquiera (por ejemplo $A \vee \neg A$). ¿Es cierto que $\pi, 1 \models \top\, \mathbf{U}\, R$?

LTL: Otros conectivos temporales

Hay otros conectivos temporales que se usan en la práctica:

$$\begin{aligned} \mathbf{F}\,\varphi &= \top\,\mathbf{U}\,\varphi, \\ \mathbf{G}\,\varphi &= \neg(\top\,\mathbf{U}\,(\neg\varphi)). \end{aligned}$$

¿Cuál es el significado de estos conectivos?

LTL y el problema de verificación

Dado: Una fórmula LTL φ

Problema de verificación: Chequear que para todos los caminos π que comienzan en el estado inicial del autómata se tiene que $\pi, 1 \models \varphi$.

¿Cuáles son las propiedades que queremos verificar en el caso del semáforo?

LTL y el problema de verificación

En el ejemplo del semáforo nos gustaría verificar la siguiente propiedad: Si el semáforo está en rojo, entonces en algún momento en el futuro va a estar en verde.

¿Cómo se expresa está propiedad en LTL?

$$\varphi \quad = \quad \mathbf{G}\,(R \to \mathbf{F}\,V).$$

¿Por qué no usamos $R \to \mathbf{F}\,V$?

¿Es cierto que $\pi, 1 \models \varphi$ para todo camino π que comienza en el estado inicial del autómata del semáforo?

Verificación: Problemas a resolver

¿Cuál es la complejidad de resolver el problema de verificación para LTL? ¿Es decidible este problema?

- ¿Existen otras lógicas para las cuales la complejidad de este problema sea menor?

¿Cuál es el poder expresivo de LTL?

- ¿Hay mejores lógicas en términos de poder expresivo?

Queremos desarrollar herramientas que nos permitan responder este tipo de consultas ...