Lógica modal computacional

Decidibilidad y complejidad

Daniel Gorín y Sergio Mera

1er cuatrimestre de 2010

Cómputos determinísticos

Programa

- Usaremos un modelo de cómputo "imperativo" (RAM)
- Podríamos haber elegido cualquier otro equivalente:
  - Máquinas de Turing
  - Cálculo lambda
  - Circuitos booleanos
  - Autómatas celulares
  - ...

Programas totales

Un programa es *total* si su ejecución termina para toda entrada.

Decidibilidad (caso determinístico)

Problema de decisión

Un *problema de decisión* es aquel cuya respuesta es "sí" ó "no".

Cómputo de problemas de decisión

Un programa ρ *computa* un problema de decisión Δ si

- ρ es total
- para toda entrada n,
  - $\Psi_\rho(n) = 1$ si $\Delta(n) =$ "sí"
  - $\Psi_\rho(n) = 0$ si $\Delta(n) =$ "no"

Problemas decidibles

Un problema (de decisión) es *decidible* si algún programa lo computa.

Cómputos no-determinísticos

La instrucción *flip*

- Introduciremos no-determinismo usando la instrucción *flip*
- Cada ejecución de *flip* devuelve 0 ó 1 (tira una moneda?)
- Podríamos haber usado otros métodos equivalentes...

Programas no-determinísticos totales

- La salida de un programa con *flip* depende de:
  - El valor de entrada
  - Lo que devuelva cada *flip* ejecutado
- Para una entrada puede haber varias ejecuciones posibles
- Un programa es *total* si termina toda ejecución de toda entrada

Decidibilidad (caso no-determinístico)

Cómputo no-determinístico de problemas de decisión

Un programa (nd.) ρ *computa* un problema de decisión Δ si

- ρ es total
- para toda entrada n,
  - Si $\Delta(n) =$ "sí", entonces **alguna** ejecución desde n da 1
  - Si $\Delta(n) =$ "no", entonces **ninguna** ejecución desde n da 1

Cómo se entiende un programa no-determinístico?

1. *Mundos paralelos.* Cada *flip* genera ejecuciones "paralelas"
2. *Magia.* Cada *flip* devuelve el "valor correcto"
3. *Siga intentando.* Cada corrida da un resultado distinto.

Decidibilidad (caso no-determinístico)

Cómputo no-determinístico de problemas de decisión

Un programa (nd.) ρ *computa* un problema de decisión Δ si

- ρ es total
- para toda entrada n,
  - Si $\Delta(n) =$ "sí", entonces **alguna** ejecución desde n da 1
  - Si $\Delta(n) =$ "no", entonces **ninguna** ejecución desde n da 1

Ejemplo. $\rho(n)$

```
i ← 2
repeat n
    i ← i + flip
return (i<n & n mod i=0)
```

- ¿Qué hace $\rho(n)$?
- ¿Es total?
- ¿Qué problema computa?
- ¿Y `return` $(1-\rho(\mathtt{n}))$? OJO!

Determinismo vs. no-determinismo

¿Se computan "más cosas" con no-determinismo?

No: se pueden escribir intérpretes (detertminísticos) de programas no-determinísticos (e.g., usando backtracking)

Se usa el modelo no-determinístico para clasificar problemas de decisión por *complejidad*.

Clases de complejidad temporal

DTIME($f(n)$) (NTIME($f(n)$))

Todos los problemas computables (no-)determinísticamente usando a lo sumo $f(n)$ pasos, con n el tamaño de la entrada

$$\mathsf{PTIME} = \bigcup_{k \geq 0} \mathsf{DTIME}(n^k)$$

$$\mathsf{EXPTIME} = \bigcup_{k \geq 0} \mathsf{DTIME}(2^{n^k})$$

$$\mathsf{2EXPTIME} = \bigcup_{k \geq 0} \mathsf{DTIME}(2^{2^{n^k}})$$

$$\mathsf{3EXPTIME} = \bigcup_{k \geq 0} \mathsf{DTIME}(2^{2^{2^{n^k}}})$$

$$\vdots$$

$$\mathsf{NPTIME} = \bigcup_{k \geq 0} \mathsf{NTIME}(n^k)$$

$$\mathsf{NEXPTIME} = \bigcup_{k \geq 0} \mathsf{NTIME}(2^{n^k})$$

$$\mathsf{2NEXPTIME} = \bigcup_{k \geq 0} \mathsf{NTIME}(2^{2^{n^k}})$$

$$\mathsf{3NEXPTIME} = \bigcup_{k \geq 0} \mathsf{NTIME}(2^{2^{2^{n^k}}})$$

$$\vdots$$

Clases de complejidad temporal

$\rho(n)$

```
i ← 2
repeat n
    i ← i + flip
return (i<n & n mod i=0)
```

- ¿Complejidad?
- ¿Qué concluimos?

$\tau(n)$

```
j ← 1, i ← 0
repeat #bits(n)
    i ← i + (flip * j)
    j ← 2 * j
return (1<i<n & n mod i=0)
```

- ¿Complejidad?
- ¿Qué concluimos?

OJO!

- Decidir si un número es compuesto está en NPTIME
- Y por lo tanto, decidir si es primo está en co-NPTIME
- Recientemente se mostró que (ambos) están en PTIME

Problemas completos

Los “difíciles” de la clase

Completos para NPTIME (aka NP-completos)

Aquellos que están en NPTIME y pueden resolver cualquier problema en NPTIME via reducciones polinomiales.

Completos para EXPTIME (aka EXPTIME-completos)

Aquellos que están en EXPTIME y pueden resolver cualquier problema en EXPTIME via reducciones polinomiales.

$\vdots$

Clases de complejidad espacial

DSPACE$(f(n))$ (NSPACE$(f(n))$)

Todos los problemas computables (no-)determinísticamente usando a lo sumo $f(n)$ bits de memoria auxiliar.

$$\mathsf{PSPACE} = \bigcup_{k \geq 0} \mathsf{DSPACE}(n^k)$$

$$\mathsf{EXPSPACE} = \bigcup_{k \geq 0} \mathsf{DSPACE}(2^{n^k})$$

$$\mathsf{2EXPSPACE} = \bigcup_{k \geq 0} \mathsf{DSPACE}(2^{2^{n^k}})$$

$$\vdots$$

$$\mathsf{NPSPACE} = \bigcup_{k \geq 0} \mathsf{NSPACE}(n^k)$$

$$\mathsf{NEXPSPACE} = \bigcup_{k \geq 0} \mathsf{NSPACE}(2^{n^k})$$

$$\mathsf{2NEXPSPACE} = \bigcup_{k \geq 0} \mathsf{NSPACE}(2^{2^{n^k}})$$

$$\vdots$$

La noción de “completitud” es análoga

Algunas relaciones entre clases de complejidad

No se sabe si son estrictas

$$\mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{NPSPACE} \subseteq \mathsf{EXPTIME} \subseteq \mathsf{NEXPTIME} \subseteq \mathsf{NEXPSPACE} \subseteq \mathsf{2EXP} \ldots$$

Se sabe que son iguales (Savitch)

$$\begin{aligned} \mathsf{PSPACE} &= \mathsf{NPSPACE} \\ \mathsf{EXPSPACE} &= \mathsf{NEXPSPACE} \\ &\vdots \end{aligned}$$

Se sabe que son estrictas (Stearns & Hartmanis; Cook)

$$\begin{gathered} \mathsf{P} \subset \mathsf{EXPTIME} \subset \mathsf{2EXPTIME} \subset \mathsf{3EXPTIME} \ldots \\ \mathsf{NP} \subset \mathsf{NEXPTIME} \subset \mathsf{2NEXPTIME} \subset \mathsf{3NEXPTIME} \ldots \\ \mathsf{PSPACE} \subset \mathsf{EXPSPACE} \subset \mathsf{2EXPSPACE} \subset \mathsf{3EXPSPACE} \ldots \end{gathered}$$

Problemas clásicos de decisión (para una lógica $\mathcal{L}$)

Model checking en $\mathcal{L}$

Dados $\mathcal{M}$ y φ, decidir si $\mathcal{M} \models_{\mathcal{L}} \varphi$

Satisfacibilidad en $\mathcal{L}$ (respecto a una clase de modelos $\mathcal{C}$)

Dada φ decidir si **existe** $\mathcal{M}$ tal que $\mathcal{M} \models_{\mathcal{L}} \varphi$ (con $\mathcal{M} \in \mathcal{C}$)

Validez en $\mathcal{L}$ (respecto a una clase de modelos $\mathcal{C}$)

Dada φ decidir si **para todo** $\mathcal{M}$ vale $\mathcal{M} \models_{\mathcal{L}} \varphi$ (con $\mathcal{M} \in \mathcal{C}$)

Satisfacibilidad y _validez_ son problemas duales

- φ es satisfacible sii $\neg\varphi$ no es válida
- φ es válida sii $\neg\varphi$ no es satisfacible

Aspectos computacionales de una lógica

Para cada uno de estos problemas vale preguntarse:

I. ¿Es decidible?

II. ¿Cuál es su complejidad de peor caso?

III. ¿Hay algoritmos que sean eficientes en el caso promedio?

Aspectos computacionales de la lógica proposicional

Model checking proposicional

I. Es decidible
II. Es lineal en la fórmula (post-order en el árbol sintáctico)
III. Es fácil de implementar de manera eficiente

Aspectos computacionales de la lógica proposicional

Satisfacibilidad proposicional

I. Es decidible
II. Está en NP (adivinar y chequear) y es completo (Cook)
III. DPLL algorithm: problemas "reales" con $> 10K$ variables

Aspectos computacionales de la lógica modal básica

Model checking (sobre modelos finitos)

I. Es decidible
II. Está en PTIME: $O(|\varphi| \cdot |W|^2)$ (eg., usando prog. dinámica)
III. Es fácil de implementar de manera eficiente

Aspectos computacionales de la lógica modal básica

Satisfacibilidad

I. Ya habíamos visto que es decidible (reducciones a FO2)

II. ??

III.

Filtraciones como cota de complejidad

Filtraciones y la propiedad de modelos finitos (repaso)

- Sea Σ un conjunto finito cerrado bajo subfórmulas
- Y sea $\mathcal{M}^f$ una filtración de $\mathcal{M}$ via Σ
- Vimos que si $\mathcal{M}, w \models \varphi$ con $\varphi \in \Sigma$, entonces $\mathcal{M}^f, |w| \models \varphi$
- Pero $\mathcal{M}^f$ es finito... cuántos estados tiene?

Corolario

- Si φ es satisfacible, tiene modelo de a lo sumo $2^{|\varphi|}$ estados
- Luego, podemos adivinar un modelo en $O(2^{|\varphi|})$
- Y podemos testear si satisface φ en tiempo polinomial
- Con lo cual, el problema seguro está en NEXPTIME

Aspectos computacionales de la lógica modal básica

Satisfacibilidad

I. Ya habíamos visto que es decidible (reducciones a FO2)

II. A lo sumo NEXPTIME (pero vamos a ver mejores cotas)

III.

to be continued