Introducció

- Definició
- Fonaments
  - Reflexió
  - Introspecció
  - Intercessió
  - Sistemes reflexius
  - Reflexió estructural
  - Reflexió de comportament

Definició

Habilitat d'un programa a examinar-se, modificar la seva estructura i el seu comportament en temps d'execusió. És l'estudi de (les conseqüències de) l'**auto-referència** en els llenguatges de programació.

Exemples d'auto-referència

Recursivitat

```
int fib(int n) {
    if (n == 0) return 0;
    if (n == 1) return 1;
    return fib(n - 2) + fib(n - 1);
}
```

Quines

Una *Quine* és un programa que s'escriu a ell mateix:

```
[ :s | Transcript show: s; show: s printString ]
    value: '[ :s | Transcript show: s; show: s printString ] value:'
```

El resultat d'executar aquest codi és:

```
[ :s | Transcript show: s; show: s printString ]
    value: '[ :s | Transcript show: s; show: s printString ] value:'
```

També es pot fer en Java:

```
public class Quine {
    public static void main(String[] args) {
        char quote = 34;
        String[] code = {
            "public class Quine {",
            "    public static void main(String[] args) {",
            "        char quote = 34;",
            "        String[] code = {",
            "        };",
            "        for (int i = 0; i < 4; i++) {",
            "            System.out.println(code[i]);",
            "        }",
            "        for (int i = 0; i < code.length; i++) {",
            "            System.out.println(quote + code[i] + quote + ',');",
            "        }",
```

```
            "        for (int i = 4; i < code.length; i++) {",
            "            System.out.println(code[i]);",
            "        }",
            "    }",
            "}",
        };
        for (int i = 0; i < 4; i++) {
            System.out.println(code[i]);
        }
        for (int i = 0; i < code.length; i++) {
            System.out.println(quote + code[i] + quote + ',');
        }
        for (int i = 4; i < code.length; i++){
            System.out.println(code[i]);
        }
    }
}
```

Halting problem (problema de l'aturada)

No existeix cap programa que, donat un programa p i uns paràmetres A et digui si $p(A)$ s'atura (es completa). Explicat d'una altra manera:

`halt(p, d) = true si p(d) s'atura i false si p(d) no s'atura`

Pot donar `true` en uns segons, però pot ser que funcioni per sempre sense aturar-se (i hauria de donar `false`, però mai ho fa perquè... I si s'atura?)

Fonaments

Reflexió

Habilitat d'un programa de modificar o manipular el que el representa mentre aquest s'executa.

Introspecció

Habilitat d'un programa d'**observar**, i per tant **raonar**, sobre ell mateix.

Intercessió

Habilitat d'un programa de **modificar**-se o d'**alterar** la seva interpetació.

Sistemes reflexius

Un sisteme refleciu...

...té una **representació interna d'ell mateix**.

...és capaç d'**actuar sobre ell mateix**.

...té capacitat d'**auto-representació** i d'**auto-modificació** continuament sincronitzades.

Reflexió estructural

Capacitat del llenguatge de programació de **reificació** completa del seu programa que s'està eecutant i dels tipus abstractes de dades.

Reflexió de comportament

Capacitat del llenguatge de programació de **reificació** completa de la seva pròpia semàntica i implementació.

> **Reïficar**: Transformar (una idea, una qualitat, una funció) en una cosa, concebre-la per analogia a l'estructura de les coses.

La *reflexió estructural* i *de comportament* determinen **quin tipus d'accés** tenim al programa, la *introspecció* i *intercessió* ens determina **què podem fer**.

Reflexió en Smalltalk