Database Integrity

How do we control methods?

Arūnas Janeliūnas

Database integrity

Ensure that the data in the database is correct at any given moment

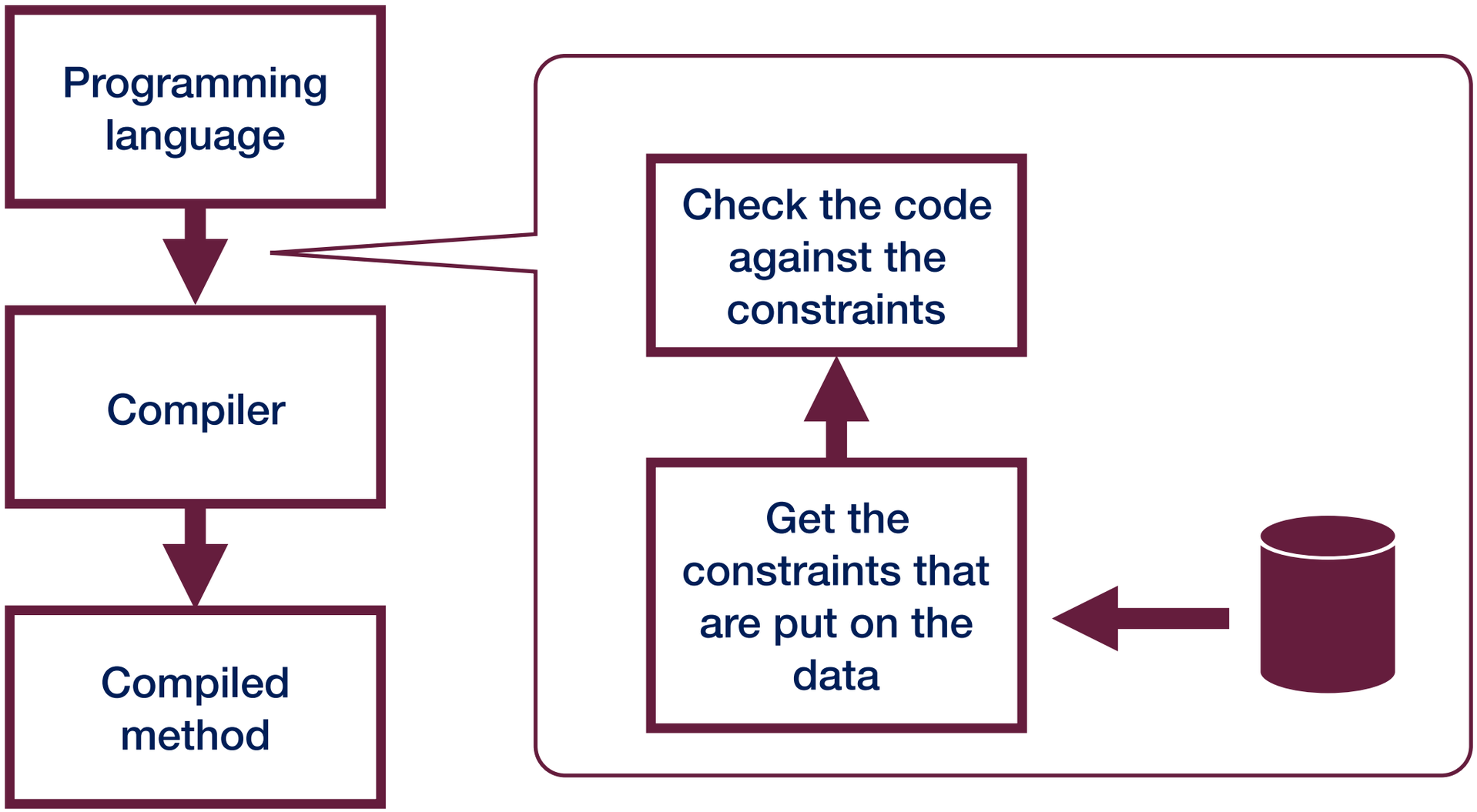
Monitoring data changes

Object methods are doing changes to the database

Monitoring methods, how are they changing the data

How we are supposed to do this?

Idea


Programming language
Compiler
Compiled method
Check the code against the constraints
Get the constraints that are put on the data

„Any“ programming language

Simplified object-oriented programming language:

```
expression ::=
    variable.attribute := variable
  | expression ; expression
  | { expression }
  | if condition then expression
  | forall variable where condition do expression
  | forone variable where condition do expression
```

Now one only need to translate his PL into the one above…

What `forone` does?

```
a.spouse.spouse := b.spouse
```

translates into…

```
forone o1 where a.spouse = o1 do
  forone o2 where b.spouse = o2 do
    o1.spouse := o2
```

Constraints language

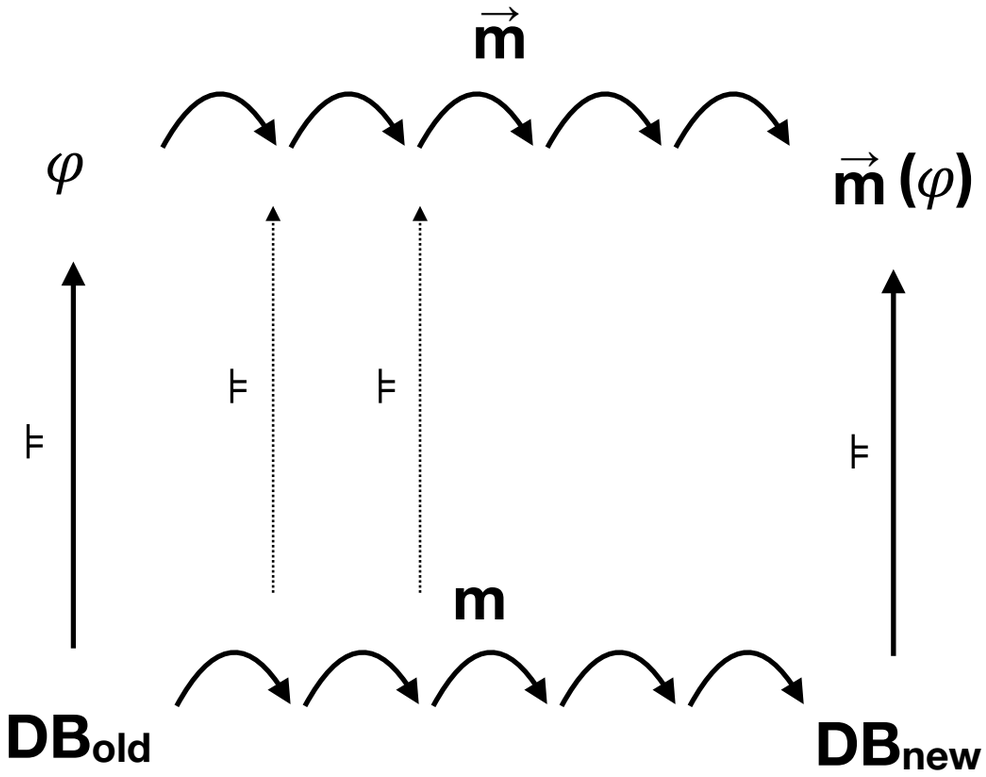
1st level logic language

```
exists x in Persons: x.spouse = nil          ∧
forall x in Persons: x.spouse ≠ x            ∧
forall x in Persons: x.spouse ∉ x.children   ∧
. . .
```

Matematikos
ir informatikos
fakultetas

Arūnas Janeliūnas

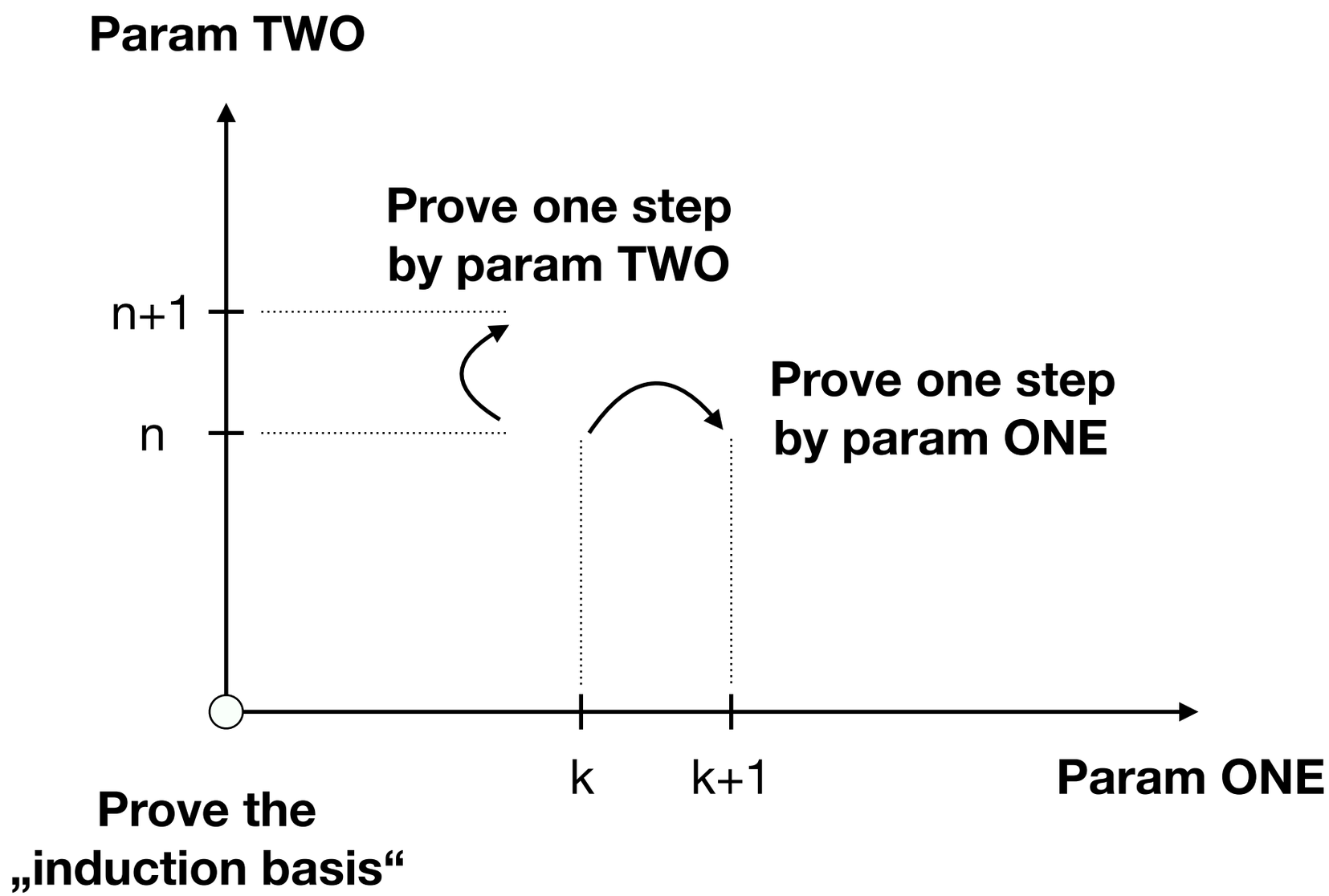
Forward predicate transformation



$\vec{m}(\varphi) \Rightarrow \varphi$

If TRUE then method is holding the constraints

Two-dimensional induction



Forward predicate transformation

Induction by the complexity of the constraint formula

1. $\vec{m}((\varphi)) \equiv \vec{m}(\varphi)$
2. $\vec{m}(\varphi \wedge \psi) \equiv \vec{m}(\varphi) \wedge \vec{m}(\psi)$
3. $\vec{m}(\varphi \vee \psi) \equiv \vec{m}(\varphi) \vee \vec{m}(\psi)$
4. $\vec{m}(\texttt{forall}\ \mathrm{x}:\ \varphi(\mathrm{x})) \equiv \texttt{forall}\ \mathrm{x}:\ \vec{m}(\varphi(\mathrm{x}))$
5. $\vec{m}(\texttt{exists}\ \mathrm{x}:\ \varphi(\mathrm{x})) \equiv \texttt{exists}\ \mathrm{x}:\ \vec{m}(\varphi(\mathrm{x}))$

Forward predicate transformation

Induction by the complexity of the method:

6. If $m \equiv \texttt{u.a:=v}$ and φ is an atomic formula:

   — If $\varphi \equiv (x.a = y)$, then $\vec{m}(\varphi) \equiv (u = x \wedge u.a = v) \vee (u \neq x \wedge u.a = v \wedge x.a = y)$

   — If $\varphi \equiv (x.a \neq y)$, then $\vec{m}(\varphi) \equiv (u = x \wedge u.a = v) \vee (u \neq x \wedge u.a = v \wedge x.a \neq y)$

   — Otherwise $\vec{m}(\varphi) \equiv \varphi \wedge u.a = v$

7. If $m \equiv i_1 \,;\, i_2$, then $\vec{m}(\varphi) \equiv \vec{i}_2(\vec{i}_1(\varphi))$

8. If $m \equiv \{i\}$, then $\vec{m}(\varphi) \equiv \vec{i}(\varphi)$

9. If $m \equiv \texttt{if}\ \psi\ \texttt{then}\ i$, then $\vec{m}(\varphi) \equiv \vec{i}(\psi \wedge \varphi) \vee (\neg\psi \wedge \varphi)$

10. If $m \equiv \texttt{forone}\ v\ \texttt{where}\ \psi(v)\ \texttt{do}\ i$, then $\vec{m}(\varphi) \equiv \texttt{exists}\ v : \vec{i}(\psi(v) \wedge \varphi)$

Forward predicate transformation

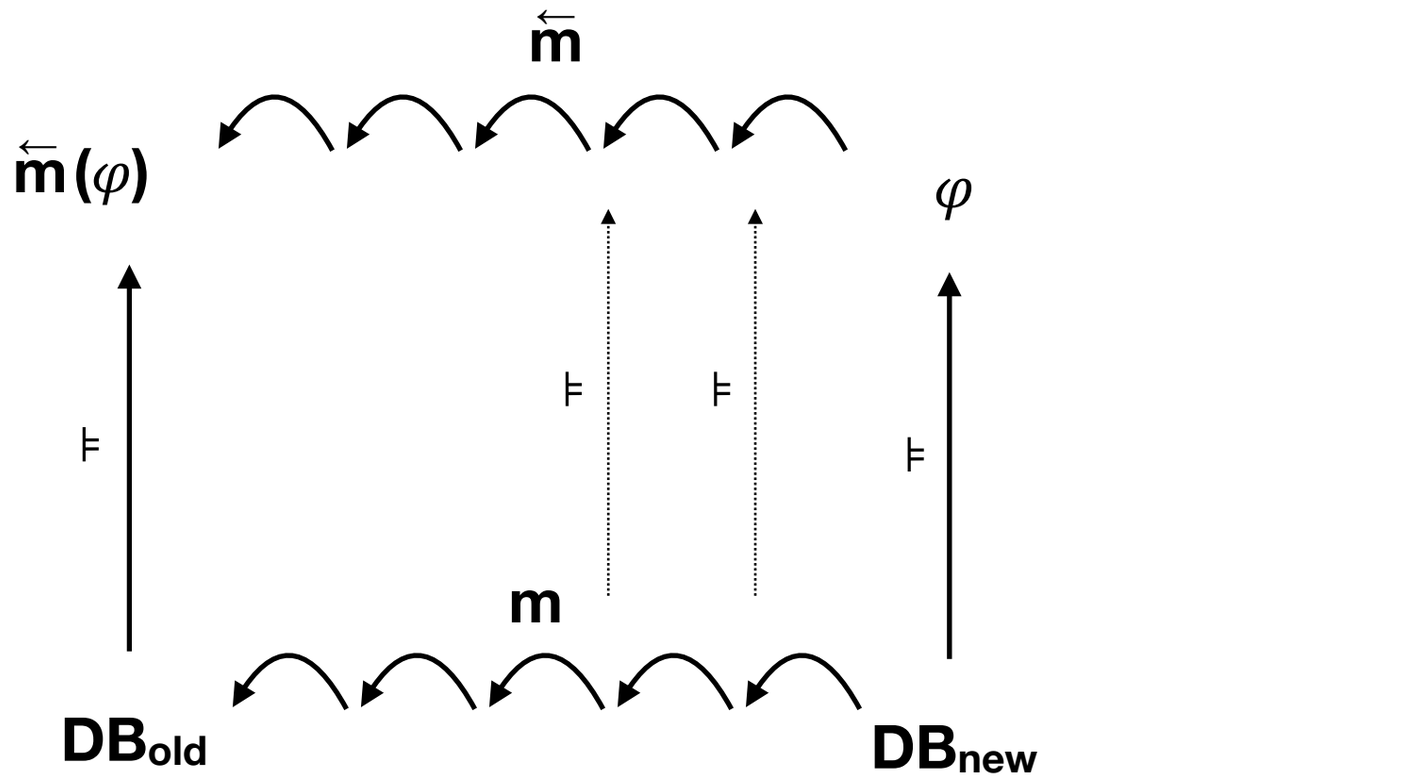
Say the method m is an instruction of the form

`forall` *variable* `where` *condition* `do` *i*

And say C is a full constraint formula. Then for any formula φ

$$\vec{\mathbf{m}}(\varphi) = \begin{cases} C & \textbf{if} \quad \varphi \Rightarrow C \quad \textbf{and} \quad \vec{\mathbf{i}}(C) \Rightarrow C \\ true & \textbf{otherwise} \end{cases}$$

Backward predicate transformation



$\overleftarrow{\mathbf{m}}(\varphi) \Rightarrow \varphi$

If TRUE then method is holding the constraints

Backward predicate transformation

Induction by the complexity of the constraint formula

1. $\overleftarrow{m}((\varphi)) \equiv \overleftarrow{m}(\varphi)$
2. $\overleftarrow{m}(\varphi \wedge \psi) \equiv \overleftarrow{m}(\varphi) \wedge \overleftarrow{m}(\psi)$
3. $\overleftarrow{m}(\varphi \vee \psi) \equiv \overleftarrow{m}(\varphi) \vee \overleftarrow{m}(\psi)$
4. $\overleftarrow{m}(\texttt{forall}\ x:\ \varphi(x)) \equiv \texttt{forall}\ x:\ \overleftarrow{m}(\varphi(x))$
5. $\overleftarrow{m}(\texttt{exists}\ x:\ \varphi(x)) \equiv \texttt{exists}\ x:\ \overleftarrow{m}(\varphi(x))$

Backward predicate transformation

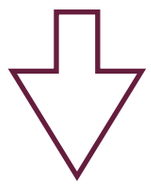
Induction by the complexity of the method:

6. If $m \equiv \texttt{u.a:=v}$ and φ is an atomic formula:
   — If $\varphi \equiv (x.a = y)$, then $\overleftarrow{m}(\varphi) \equiv (u = x \wedge v = y) \vee (u \neq x \wedge x.a = y)$
   — If $\varphi \equiv (x.a \neq y)$, then $\overleftarrow{m}(\varphi) \equiv (u = x \wedge v \neq y) \vee (u \neq x \wedge x.a \neq y)$
   — Otherwise $\overleftarrow{m}(\varphi) \equiv \varphi$
7. If $m \equiv i_1 ; i_2$, then $\overleftarrow{m}(\varphi) \equiv \overleftarrow{i_1}(\overleftarrow{i_2}(\varphi))$
8. If $m \equiv \{i\}$, then $\overleftarrow{m}(\varphi) \equiv \overleftarrow{i}(\varphi)$
9. If $m \equiv \texttt{if}\ \psi\ \texttt{then}\ i$, then $\overleftarrow{m}(\varphi) \equiv (\psi \wedge \overleftarrow{i}(\varphi)) \vee (\neg\psi \wedge \varphi)$
10. If $m \equiv \texttt{forone}\ v\ \texttt{where}\ \psi(v)\ \texttt{do}\ i$,
    then $\overleftarrow{m}(\varphi) \equiv (\texttt{exists}\ v : \psi(v)) \wedge \overleftarrow{i}(\varphi)$

Backward predicate transformation

Good news - it allows method auto-correction

```
method m (params) in class K
{
        method_body
}
```



```
method m (params) in class K
{
    if( ←m(C) )
      method_body
}
```